Cours : Merise

Thème : modélisation des données avec les extensions Merise 2

Notions abordées :
- Concepts de base de modélisation de données Merise : entités, associations, propriétés, propriétés portées, cardinalités, CIF, CIM, etc. ;
- Concepts avancés de modélisation de données avec les extensions Merise 2 : héritage ou spécialisation, contraintes d'association, réflexivité, etc. ;
- Transcription d'un modèle conceptuel en schéma relationnel.

1. Merise - rappels

1.1. MCD

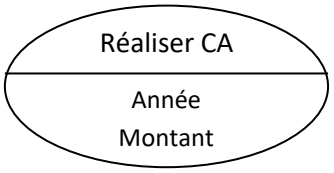
MCD (Modèle Conceptuel des Données) : un MCD est un diagramme permettant de donner une représentation schématique de tout ou partie d'une base de données relationnelle. Lorsqu'une base de données devient consistante, on a tendance à diviser le modèle en sous-modèles. Les MCD font partie plus généralement d'une méthode de conception d'origine française appelée la méthode MERISE, laquelle définit un certain nombre de schémas et diagrammes. MCD, MOT et schéma relationnel sont certainement les plus importants.
Le MCD vise quant à lui à modéliser des tables et les relations existants entre elles. De manière quelque peu grossière, on peut voir une table comme un « gros tableau ».
Avec l'apparition puis le développement de ce qu'on qualifie de programmation orientée objets (abrégée POO ou OOP en anglais), cette méthode de conception franco-française tend depuis bien des années à être remplacée en pratique par la notation UML (*Unified Modeling Language*). On y retrouve des concepts similaires. Cependant, celle-ci fournit, disons, une vision plus orientée programmeur.

MCD et syntaxe : le MCD possède une syntaxe et un vocabulaire qui lui sont propres. On parle d'entités et d'associations. C'est pourquoi le MCD est parfois appelé modèle entités-associations.

Entité : une entité une unité élémentaire qui se suffit à elle-même (exemples : client, fournisseur, produit, article…). A toute entité correspond une table (grossièrement, un tableau). Elle possède un nom et des propriétés, encore appelés des attributs (grossièrement, les colonnes du tableau). Optionnellement, comme en algorithmique, on précise le type, à savoir le type de chaque propriété (qu'importe !) La ou les propriétés soulignées est ou sont appelées clef primaire.	Une entité est représentée par un rectangle. **Client** N° Client — Integer Raison sociale — Varchar(30) Date immatriculation — Date …. — …

Association : une association décrit une relation existant entre plusieurs entités. Elle n'a de sens que sous réserve de l'existence des entités dont elle fait la relation. On entend par « relation » comme une forme de relation d'appartenance (exemple : un client a 0 ou plusieurs produits, sous-entendu il commercialise 0 à plusieurs produits).
Elle possède un nom et éventuellement des propriétés, appelées propriétés portées (sous-entendu portées par l'association).
Important ! A une association correspond ou non une table. Si l'association correspond à une CIF, l'association ne correspond pas à une table mais à une clef étrangère. Si l'association correspond à une CIM, elle correspond à une table.

Une association est représentée par un ovale.



- *L'association ci-dessus a 2 propriétés portées.*
- *Le nom d'une association est souvent un verbe quoique cela n'ait rien d'obligatoire.*
- *Essentiellement, l'utilisation d'un verbe facilite la lecture du diagramme.*

Clef primaire : si une table est une sorte de tableau et les propriétés des sortes de colonnes, alors une clef primaire correspond grossièrement à l'identifiant de la ligne, i.e. la ou les colonnes identifiants la ligne de manière unique. Toute table possède une clef primaire, un identifiant. Cet identifiant est unique, il est propre à chaque ligne. Lorsque la clef primaire est constituée de plusieurs propriétés, on parle de clef primaire composée (sous-entendu composée de plusieurs propriétés). Les autres propriétés d'une ligne sont en quelque sorte rattachées à cette clef primaire. On parle de dépendance fonctionnelle.

Important : la clef primaire propre à une entité doit figurer dans les propriétés de l'entité. Elle est soulignée. La clef primaire correspondant à une association est sous-entendue. Elle est constituée de la clef primaire de chacune des entités qu'elle met en relation.

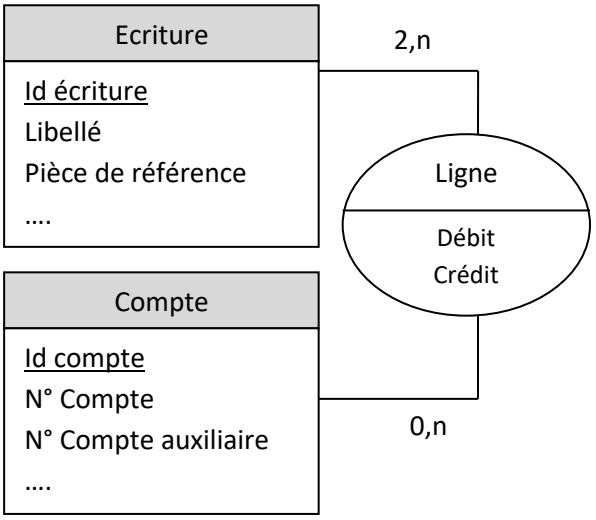
Cardinalités : une association relie au moins deux entités entre elles. Si elle relie trois entités, on dira que c'est une association ternaire, ou plus simplement une ternaire.
Les cardinalités précisent le lien entre les entités reliées. Elles permettent de préciser la quantité minimale et maximale qu'une entité a d'une autre entité.
Dans le cas d'une ternaire, les cardinalités seront forcément des 0,n ou plus généralement de la forme a,n avec a un entier quelconque.

Important !

Lorsque les cardinalités sont de la forme :
- **X,n - Y,n** (exemple : 0,n - 1,n) avec X et Y des nombres, on parle de CIM (contrainte d'intégrité multiple) ;
- **X,1 - Y,n** (exemple : 1,1 - 0,n) avec X et Y des nombres, on parle de CIF (contrainte d'intégrité fonctionnelle).

Si une association représente une CIM, alors

Association et entités mises en relation sont reliées par un trait, appelée patte. Sur ces pattes, on précise la cardinalité, et optionnellement le rôle de la patte, i.e. sa signification.



A droite, les cardinalités signifient :

l'association correspond à une table. Si une association représente une CIF, alors elle ne correspond pas à une table mais à une clef étrangère. La clef étrangère apparaît alors dans la table correspondant à l'entité ayant la cardinalité de la forme X,1.	○ *Une écriture comptable est constituée de 2 à plusieurs lignes.* ○ *Un compte peut être apparu dans aucune à plusieurs écritures comptables.* ○ *L'association « Ligne » représente une CIM. Ligne correspond donc à une table.*

Clef étrangère : dans un MCD, une clef étrangère est représentée par une CIF. Dans un schéma relationnel, la clef étrangère est représentée par un ou plusieurs champs. Lorsque la clef étrangère est constituée de plusieurs propriétés, on parle de clef étrangère composée. Grossièrement, une clef étrangère correspond, dans une table, pour chaque ligne, à une valeur pointant vers l'identifiant d'une ligne d'une autre table.

1.2. Schéma relationnel - généralités

Le schéma relationnel, forme de Modèle Logique des Données (MLD), est une autre représentation d'une base de données relationnelle, plus concrète (moins « conceptuelle »). Il représente concrètement la structure des données, i.e. les tables, leur clef primaire, leurs clefs étrangères et leurs autres champs. On le rédige sous la forme d'une sorte de liste dans laquelle figurent les informations citées ci-avant.

La syntaxe : **MaTable(MaClefPrimaire, #UneClefEtrangere, UnChamp, …)**

Détail :
- Le ou les champs constituant la clef primaire sont soulignés.
- Les clefs étrangères éventuelles sont préfixées par un « # ».
- Les autres champs sont simplement mentionnés.
- La clef primaire peut être constituée de plusieurs champs, tous soulignés.
- Il peut y avoir aussi bien aucune qu'une ou plusieurs clefs étrangères.
- Il peut très bien n'y avoir aucune, une ou plusieurs clefs étrangères dans la clef primaire.
- Une clef étrangère peut très bien être constituée de plusieurs champs, la syntaxe diffère alors légèrement :
  MaTable(MaClefPrimaire, #(Champ1, Champ2), UnChamp, …)
  Le couple (Champ1,Champ2) fait ci-dessus office de clef étrangère composée.

Exemple : on notera que la CIM (« Ligne ») correspond bien à une table.

Ecriture(EcritureNum, EcritureLib, PieceRef, …)
Clef primaire : EcritureNum
Ligne(#EcritureNum, #CompteId, Debit, Credit, Lettrage, …)
Clef primaire : EcritureNum, CompteId
Clefs étrangères :
- EcritureNum en référence à Ecriture(EcritureNum)
- CompteId en référence à Compte(CompteId)

Compte(CompteId, CompteNum, CompteAuxNum, CompteLib, …)

2. Merise 2 - extensions Merise et entités

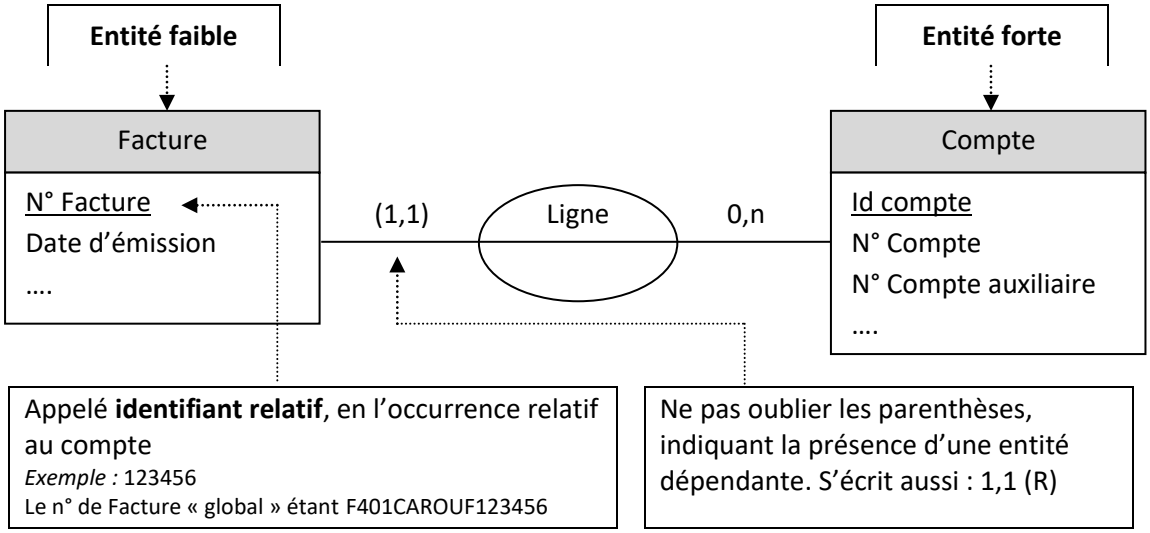
2.1. Entité dépendante et identifiant relatif

Quoiqu'une entité soit une unité *a priori* indépendante, il arrive qu'on veuille la faire dépendre d'une ou plusieurs aux entités. En ce sens, une telle entité s'apparente à une association. Il y a principalement deux cas pratiques où l'on est amené à faire dépendre une entité d'une ou plusieurs autre(s) :

Cas n° 1 : lorsque l'on souhaite que l'identifiant d'une entité dépende d'une autre entité.

Exemple : facture n°F401CAROUF123456

On a choisi de faire dépendre notre n° de facture du compte auxiliaire propre à carrefour (401CAROUF) et de préciser le ensuite la « combientième » facture est adressée à carrefour. Le 123456 correspond à la valeur de l'identifiant relatif, à savoir que c'est la 123456ème facture adressée spécifiquement à Carrefour. Une telle situation peut être modélisée par une entité dépendante. On a alors besoin de connaître le compte auxiliaire (pour retrouver la valeur 401CAROUF) et le « n° de facture » partiel relativement à Carrefour.



N.B. : l'identifiant relatif est symbolisé par **(1,1)**, i.e. une CIF entre parenthèses, ou par **1,1 (R)**.

Facture(#IdCompte, NumFacture, DateEmission, …)
  Clef primaire : IdCompte, NumFacture
  Clef étrangère : IdCompte en référence à Compte(IdCompte)
Compte(IdCompte, NumCompte, NumCompteAux, …)
  Clef primaire : IdCompte

Cas n°2 : lorsque l'existence d'une entité n'a de sens que sous réserve de l'existence d'une ou plusieurs autres.

Exemple : un appartement n'existe que sous réserve de l'existence de l'immeuble. Détruisez l'immeuble,

l'appartement n'existe plus. On peut alors modéliser cette dépendance par une entité dépendante.

Immeuble(#NumImmeuble, Adresse, CodePostal, Ville, …)
Clef primaire : NumImmeuble
Appartement(#NumImmeuble, NumAppartement, Superficie, …)
Clef primaire : NumImmeuble, NumAppartement
Clef étrangère : NumImmeuble en référence à Immeuble(NumImmeuble)

Important ! Comprenez bien que, lorsque l'on modélise une base de données, on pense la modélisation en fonction du résultat qu'on veut obtenir. Il y a souvent plusieurs possibilités. On choisit (doit choisir) le modèle qui semble le mieux répondre au besoin.

2.2. Pseudo-entité (ou agrégat)

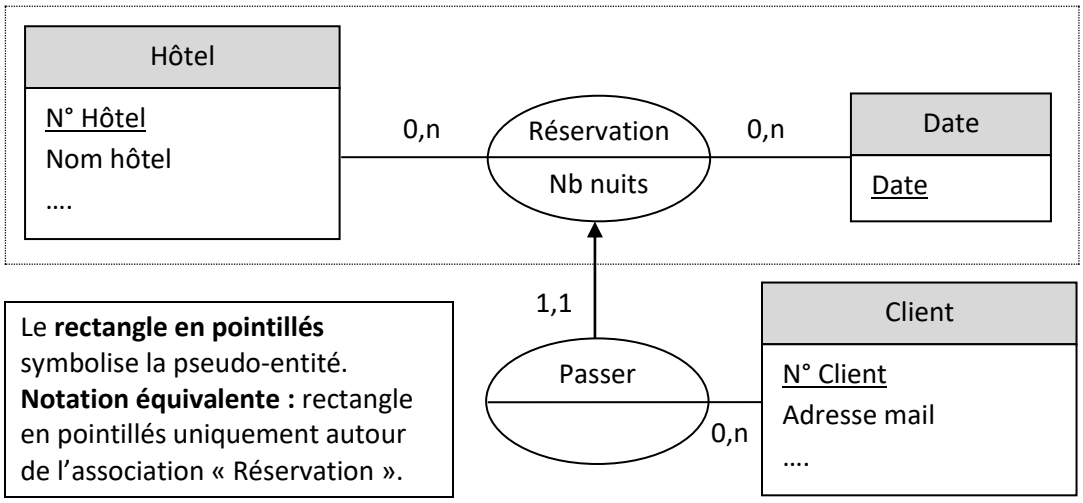
Le concept de pseudo-entité intervient dès lors que l'on souhaite qu'une association se comporte comme une entité. C'est pourquoi l'on emploiera volontiers le terme de pseudo-entité, synonyme du terme « agrégat ».

Il advient en effet que l'on souhaite mettre en relation une association avec une ou plusieurs entités. Autrement dit, il arrive que l'on souhaite pouvoir adjoindre des CIF (des clefs étrangères quoi…) à une association. Dit autrement : on peut avoir une association de type CIM, relative à deux entités (ou plus), et vouloir lui ajouter des CIF (comme on le ferait pour une entité). Alors, on peut modéliser cette situation grâce à une pseudo-entité.

Comment se visualiser une pseudo-entité ? Se dire, une pseudo-entité, voilà, c'est une CIM centrale (i.e. typiquement une association relative à deux entités) + une ou plusieurs CIF associées à la CIM.

Quand est-on (doit-on) être tenté d'utiliser une pseudo-entité ? Lorsque l'on souhaite mettre en relation une ou plusieurs entités et une association.

Exemple : on souhaite réserver une nuit d'hôtel à une date donnée. La réservation n'a de sens que relativement à l'hôtel et à la date (i.e. réservez d'une part l'hôtel, d'autre par la date, ne fait pas sens…).



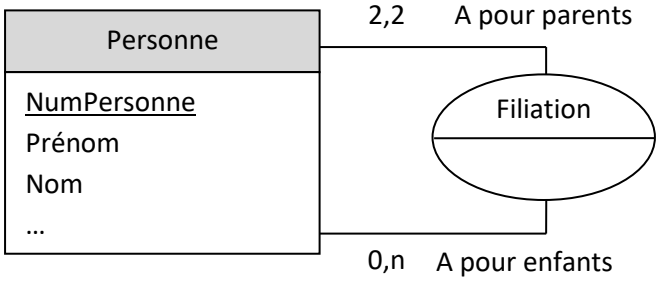
Important ! Lorsqu'il existe une dimension temporelle (date, période, etc.), on modélise le temps au moyen d'une entité.

Schéma relationnel : notez que, très normalement, « Réservation » est une CIM et se comporte comme une CIM, « Passer » est une CIF et se comporte également comme une CIF.

Hôtel(NumHotel, NomHotel, …)
Clef primaire : NumHotel
Date(Date) ← en pratique, inutile de créer une table date
Clef primaire : Date
Client(NumClient, MailClient, …)
Clef primaire : NumClient
Réservation(#Hotel, #Date, #Client, NbNuits) ← en pratique, la date n'est pas une clef étrangère
Clef primaire : Hotel, Date
Clefs étrangères :
- Hotel en référence à Hotel(NumHotel)
- Date en référence à Date(Date)
- Client en référence à Client(NumClient)

2.3. Réflexivité

En matière de bases de données, la réflexivité est un concept nécessaire afin de produire une structure arborescente, une structure_récursive. En d'autres termes, la réflexivité permet de modéliser des hiérarchies, des relations_d'antécédence. Pour exemple, l'on ne veut prendre que celui de l'arbre généalogique. Un arbre généalogique représente des liens de filiations entre personnes. Alors pourquoi aurait-on besoin d'une autre entité que l'entité personne ?
Avec la réflexivité, on retrouve une fois encore les concepts de CIF et de CIM. Seulement, au lieu d'avoir deux et plus entités reliées entre elles, l'on a une entité reliée à elle-même. Cette fois-ci, on est contraint de préciser le rôle de chacun des pattes de l'association_réflexive. Dans un arbre généalogique, une personne a le rôle du parent, l'autre de l'enfant. Les deux sont bien des personnes. Elles n'ont juste pas le même rôle. C'est bien comme s'il y avait deux entités, mais deux fois la même.



Comment s'imaginer ça ? On peut se dire que tout se passe comme s'il y avait deux tables : la table « Parent » (patte « A pour parents ») et la table « Enfant » (patte « A pour enfants »). Mais on a réuni les deux tables dans une seule.

Attention ! C'est une CIF avec une cardinalité 2,2. Elle se comporte donc comme deux CIF 1,1. A savoir, cela fait 2 clefs étrangères.

Schéma relationnel : on notera que « Filiation » correspond à une CIF, « Filiation » ne correspond donc pas à une table, mais, en l'occurrence à deux clefs étrangères (cardinalité 2,2 et non 0,1 ou 1,1).

Personne(NumPersonne, #Pere, #Mere, Prenom, Nom, …)
Clef primaire : NumPersonne
Clefs étrangères :
- Pere en référence à Personne(NumPersonne)
- Mere en référence à Personne(NumPersonne)

Ci-dessus, les champs « Pere » et « Mere » sont chacun une clef étrangère pointant vers une personne. Qu'il s'agisse de parents ou d'enfants, il n'y a finalement ici que des personnes !

Requête utilisant la réflexivité : nombre d'enfants de chaque personne. On notera qu'on est obligé d'utiliser deux fois la table « Personne », qu'on est ainsi obligé d'utiliser des alias.

```
SELECT Parent.Prenom, Parent.Nom, count(NumPersonne) AS [ Nb enfants ] ← on souhaite le nombre de
FROM Personne AS Parent LEFT JOIN Personne AS Enfant        ← Tous les parents/enfants
ON Enfant.Pere = Parent.NumPersonne                         ← Relie le père à son enfant (s'il en a)
OR Enfance.Mere = Parent.NumPersonne        ← Ou relie la mère à son enfant (si elle en a)
GROUP BY Parent.NumPersonne         ← On souhaite le nombre d'enfants par personne/parent
```

N.B. : on aurait aussi bien pu mettre count(*).

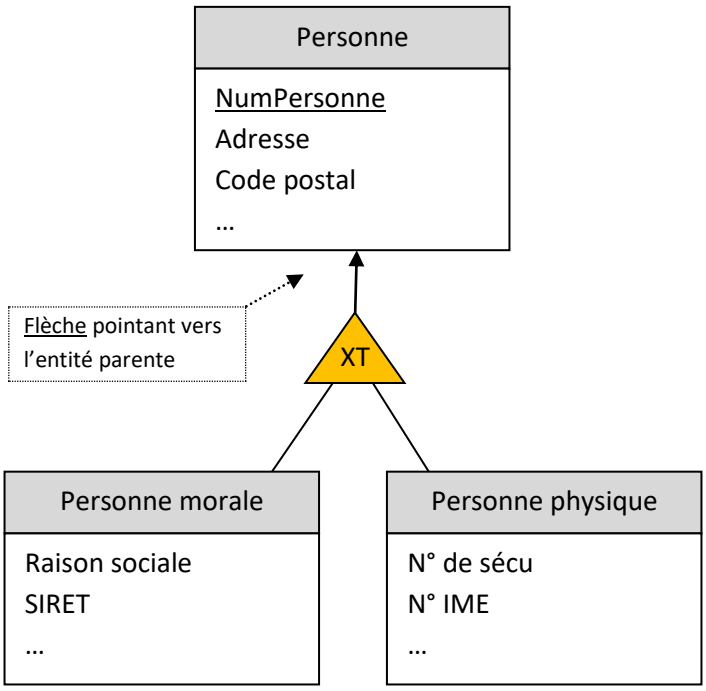
Comment se visualiser une association réflexive ? Se dire, lorsque j'ai besoin de placer une CIF ou une CIM entre deux entités A et B, cela pose pas de problème. Alors, il n'y aucune raison que cela pose un souci de placer une CIF ou une CIM entre l'entité et A et l'entité A.

Quand est-on (doit-on) être tenté d'utiliser association réflexive ? Dès lors que l'on a besoin de mettre en relation une entité avec elle-même.

2.4. Spécialisation (ou héritage)

La spécialisation est un concept également appelé héritage ou encore généralisation, les deux dernières dénominations nous venant entre autres de la POO et de l'UML. En ce qui concerne les bases de données relationnelles, La spécialisation consiste à regrouper les propriétés communes d'entités semblables au sein d'une même entité, appelée entité générique (également : entité parente ou sur-type). Les entités semblables sont appelées entités spécialisées (également : entités filles ou sous-type). Il a plusieurs types de spécialisation : **rien** (vide), **X**, **T**, **XT** ou **+**.
Il n'y a en théorie qu'une façon de modéliser la spécialisation sous forme de MCD (au moyen d'un triangle reliant les entités filles à l'entité parente). On rencontre cependant quelques variantes. En revanche, il y a en pratique plusieurs manières (essentiellement 3) de modéliser la spécialisation sous forme de schéma relationnel, chacune d'entre elles présentant des avantages et des inconvénients.

Quand est-on (doit-on) être tenté d'utiliser une pseudo-entité ? Lorsque l'on a besoin de deux entités, lorsque ces entités sont similaires et ne diffèrent que par quelques propriétés ou ont beaucoup de propriétés communes (i.e., ces entités sont sémantiquement proches).



Les types de spécialisation :

o **X** (exclusivité) : équivaut à un OUX (XOR) logique (ou exclusif).
Description : on est l'une, l'autre ou aucune, mais pas les deux à la fois.

o **T** (totalité) : équivaut au OU logique.
Description : on est l'une, l'autre, les deux, mais par aucune des deux.

o **XT** ou **+** (partition) : totalité + exclusivité
Description : on est l'une, l'autre (et donc ni les deux, ni aucune des deux).

o **vide** : tout et n'importe quoi
Description : on est l'une, l'autre, les deux ou aucune des deux.

Par l'une, l'autre… on entend : l'une, l'autre… des entités filles.

Schéma relationnel : notez que, dans le MCD, les clefs primaires des entités filles sont sous-entendues. Il n'y a pas lieu de les préciser dans le MCD ! Ci-dessous, les trois modélisations possibles sous forme de schéma relationnel.

Variante à 1 entité : toutes les propriétés au sein d'une unique table correspondant à l'entité parente

Personne(NumPersonne, Type, Adresse, CodePostal, RaisonSociale, SIRET, NumSecu, NumIME)
Clef primaire : NumPersonne
Commentaire : le champ « Type » est appelé champ ou propriété discriminante. Il permet de déterminer si la personne est une personne physique ou moral. Il acceptera par exemple les valeurs « PM » et « PP ».

Variante à 2 entités : deux tables distinctes (une par entité fille) en répétant les propriétés communes

PersonneMorale(#NumPersonne, Adresse, CodePostal, RaisonSociale, SIRET, …)
Clef primaire : NumPersonne
PersonnePhysique(#NumPersonne, Adresse, CodePostal, NumSecu, NumIME, …)
Clef primaire : NumPersonne

Variante à 3 entités : trois entités distinctes (une par entité)

Personne(NumPersonne, Type, Adresse, CodePostal, …)
Clef primaire : NumPersonne
PersonneMorale(#NumPersonne, RaisonSociale, SIRET, …)
Clef primaire : NumPersonne
Clef étrangère : NumPersonne en référence à Personne(NumPersonne)
PersonnePhysique(#NumPersonne, NumSecu, NumIME, …)
Clef primaire : NumPersonne

Clef étrangère : NumPersonne en référence à Personne(NumPersonne)
Commentaire : le champ « Type » est une fois encore une propriété discriminante. On notera que celle-ci est cette fois-ci optionnelle mais facilement utilisée en pratique. En effet, cette propriété permet de déterminer, à partir de la table « Personne », de savoir immédiatement si le reste des informations relatives à une personnes sont présentes dans la table « PersonneMorale » ou « PersonnePhysique ».

Commentaire :

- La troisième variante est la plus proche du MCD correspondant. Cependant, l'obtention de toutes les informations relatives à chaque personne nécessite une jointure ce qui implique un temps de calcul plus élevé (complexité algorithme plus élevée : $O(n \times log(n))$).
- La première variante peut être vue comme étant peu élégante mais permet d'obtenir toutes les informations relatives aux personnes en parcourant une seule table. Ceci induit un temps de calcul moindre (complexité algorithmique : $O(n)$) .
- Finalement, il convient en général d'éviter la deuxième variante, très vite source d'ennuis.

Requêtes utilisant la spécialisation (3ème variante) :

- Liste des personnes morales :

```
SELECT * FROM Personne NATURAL JOIN PersonneMorale
SELECT * FROM Persone AS P INNER JOIN PersonneMorale AS M ON P.NumPersonne = M.NumPersonne
```

- Toutes les personnes (cas typique d'utilisation de l'opérateur ensembliste UNION) :

```
SELECT NumPersonne, RaisonSociale, SIRET, '' '', '' '' FROM PersonneMorale
UNION
SELECT NumPersonne, '' '', '' '', NumSecu, NumIME FROM PersonnePhysique
```

3. Merise 2 - extensions Merise et contraintes

3.1. Contraintes d'associations

Les CIF et les CIM ne sont pas les seules contraintes pesant sur une base de données relationnelle. Il en existe de nombreuses. On parle de contraintes, ou plus exactement de contraintes d'intégrité référentielle, car, lorsque l'on crée une base de données, on cherche à pouvoir stocker des données intègres et conformes au référentiel. Cela signifie que l'on cherche à avoir des données conformes au bon sens et aux règles de gestion imposées.

Il est par exemple absurde que l'on puisse ajouter dans un plan comptable le compte « 101BIGUP » ou « 123456789123456 » car : on ne peut pas créer de compte auxiliaire (101BIGUP) sur un compte de classe 1 ; tout n° de compte comporte au maximum 13 caractères (123456789123456, 15 caractères).

Certaines contraintes, à l'instar de celles citées précédemment, ne peuvent guère être modélisées sur un MCD ou un schéma relationnel. Néanmoins, elles peuvent effectivement être implémentées (mises en œuvre) directement sur une base de données relationnelle pour la plupart ou peuvent toutes être

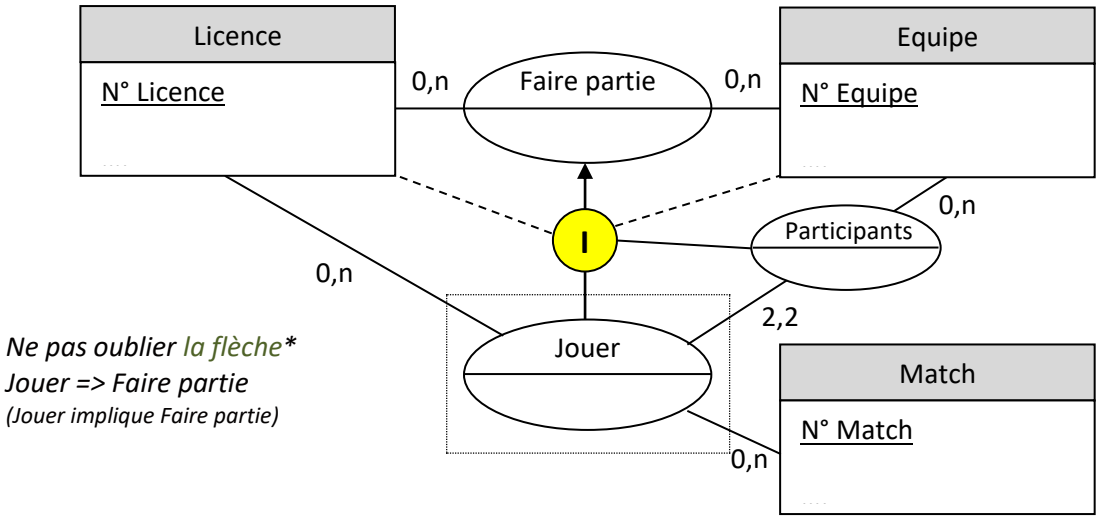
implémentées sur un logiciel ayant recours à une base données (on parle de contrôle applicatif). En particulier, les déclencheurs (triggers) permettent en outre d'effectuer des traitements de contrôle des données lors d'opérations d'insertion, de modification ou encore de suppression au sein de la base de données. Autrement dit, il s'agit de procédures/fonctions déclenchées (fonctions de rappel, *callback*) lors d'un évènement sur la base de données. Il feront l'objet d'un autre cours.

En revanche, il existe encore une catégorie de contrainte que l'on peut modéliser sur un MCD : les contraintes d'associations. Les contraintes d'association sont sans impact sur le schéma relationnel. Le contrôle d'intégrité référentiel lié à ces contraintes peut être effectué au moyen de triggers.

Contrainte d'inclusion : le terme « inclusion » est peu intuitif, mais cohérent. Comme les autres contraintes d'associations, la contrainte d'inclusion est une contrainte ensembliste. Une telle contrainte symbolise une « implication ». Elle représente le fait que, pour que l'on participe à une relation, il faille qu'on apparaisse dans une autre relation.

Autrement dit, cela représente la contrainte suivante : on souhaite qu'une ligne ne puisse apparaître dans la table correspondant à une association que si tout ou partie de sa clef primaire apparaît dans la table correspondant à une autre association.

Dans l'exemple ci-dessous, la contrainte d'inclusion représente le fait qu'un sportif ne puisse jouer un match que s'il fait partie de l'une des équipes s'affrontant. Ce qui signifie : pour qu'une couple (N°Licence, N°équipe) apparaissent dans la table « Jouer », il faut que ce même couple apparaisse dans la table « Faire partie ».



N.B.: les traits en pointillés sont appelés les pivots. Le ou les pivot(s) pointent les entités dont la clef primaire doit figurer à la fois dans la ou les association(s) « sources » et dans l'association de « destination ».

* La contrainte d'inclusion est orientée.

Contrainte d'exclusion « X » (exclusivité) : c'est le même « X » que celui de la spécialisation. Une contrainte d'exclusion, placée entre deux associations, signifie qu'on ne puisse pas participer aux deux relations à la fois (l'un, l'autre ou aucune).

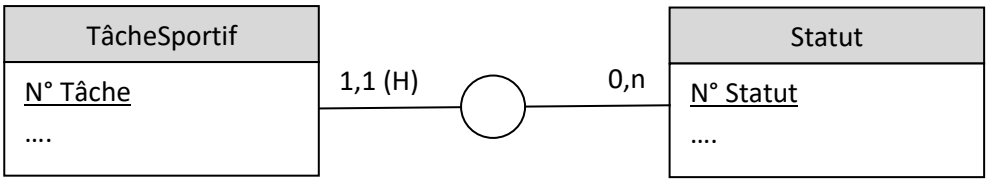
Contrainte de totalité « T » (totalité) : c'est le même « T » que celui de la spécialisation. Une contrainte de totalité, placée entre deux associations, signifie qu'on doit absolument participer à l'une ou l'autre des deux associations (l'une, l'autre ou les deux).
Contrainte de partition « XT » ou « + » (partition) : ce sont les mêmes « XT » et « + » que ceux de la spécialisation. Une contrainte de partition, placée entre deux associations, signifie qu'on ne puisse ni participer aux deux relations à la fois ni à participer à aucune (l'une, l'autre).
Contrainte de partition « = » (simultanéité) : Une contrainte de simultanéité, placée entre deux associations, signifie qu'on doive absolument (simultanément) participer aux deux associations. Cela revient à une sorte de « ET » logique (voir fiche algorithmique).

3.2. Contraintes d'historisation

L'historisation est une autre problématique habituelle de logique métier. Il advient que l'on veuille connaître un historique et que cet historique corresponde à l'évolution des données dans le temps. On parle alors d'historisation. Cela se traduit par la nécessité de créer une table d'historique correspondant à une table dont on souhaite connaître l'évolution de certaines données au travers du temps. Autrement dit, une tûple (un enregistrement) a alors un à plusieurs états distincts dans le temps, ou encore, si on pense objets, un objet a plusieurs états dans le temps.

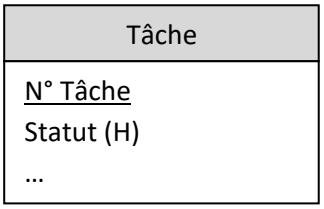
En Merise, l'historisation est symbolisée par : **(H)**. L'historisation peut porter sur une CIF (clef étrangère) et/ou une ou plusieurs propriétés d'une entité et/ou une entité toute entière.

Historisation d'une CIF : on mémorise l'évolution d'un statut dans le temps.



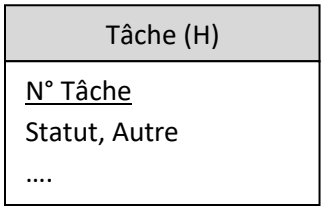
Tâche(NumTache, …)
Statut(NumStatut, …)
TâcheHistorique(NumTache, NumStatut, NumEtat)

Historisation d'une CIF : on mémorise l'évolution d'un statut dans le temps.



Tâche(NumTache, …)
TâcheHistorique(NumTache, NumEtat, Statut)

Historisation de toute une table : on mémorise l'évolution de la tâche dans le temps.



Tâche(NumTache, …)
TâcheHistorique(NumTache, NumEtat, Statut, Autre, …)

N.B. : on peut bien entendu enrichir les tables d'historique avec un champ DateEtat afin de savoir quand le nouvel état est apparu.

3.3. Contraintes de stabilité

Un champ, une clef étrangère ou un enregistrement stable (immutable) est un enregistrement qui ne peut être modifié une fois précisé, une fois fixé. C'est-à-dire que, dès une opération d'insertion effectuée, toute opération de modification effectuée sur le champ stable, la clef étrangère stable ou l'enregistrement stable est impossible. A la manière de l'historisation, la contrainte de stabilité est symbolisée par : **(S)**.