Initiation à la conception de bases de données relationnelles avec MERISE

Par Idriss NEUMANN

Date de publication : 28 février 2012

Dernière mise à jour : 17 octobre 2012

TOUT PUBLIC

Ce cours est conçu pour ceux qui souhaitent s'initier rapidement à la conception d'une base de données relationnelle à l'aide de la méthode d'analyse MERISE. Il est en rapport direct avec le programme de certaines formations d'études supérieures comme le BTS Informatique de Gestion ou encore le DUT informatique.

Vous pouvez commenter l'article en suivant ce lien : **Commentez**

I - MERISE au service des systèmes d'information....3
I-A - Le système d'information....3
I-B - MERISE....3
II - Modélisation d'une base de données au niveau conceptuel....3
II-A - Les règles de gestion métiers....4
II-B - Le dictionnaire des données....4
II-C - Les dépendances fonctionnelles....6
II-D - Le Modèle Conceptuel de Données (MCD)....7
II-D-1 - Les entités....7
II-D-2 - Les associations....8
II-D-3 - Élaboration du MCD....10
III - Modélisation d'une base de données au niveau logique et passage au SQL....11
III-A - Le passage du MCD au MLD et SQL....11
III-A-1 - Les relations....11
III-A-2 - Règles de conversion....12
III-A-2-a - Règle 1 - conversion d'une entité....12
III-A-2-b - Règle 2 - conversion d'associations n'ayant que des cardinalités de type 0/1,N....12
III-A-2-c - Règle 3 - conversion des associations ayant au moins une cardinalité de type 1,1....13
III-A-2-d - Règle 4 - conversion des associations ayant au moins une cardinalité de type 0,1 (et dont les autres cardinalités sont de type 0,1/N)....14
III-A-3 - Élaboration du MLD et passage au SQL....15
III-B - Règles de vérification des niveaux de normalisation....17
III-C - Cas particuliers....17
III-C-1 - Les associations réflexives....17
III-C-2 - Règle de conversion exceptionnelle pour certaines entités simples....19
IV - Les extensions apportées par MERISE II....19
IV-A - L'identification relative....19
IV-B - L'héritage et ses limites....20
IV-B-1 - L'héritage par disjonction (ou exclusion)....22
IV-B-2 - L'héritage par couverture (ou totalité)....23
IV-B-3 - L'héritage par partition (totalité et exclusion)....24
IV-B-4 - Passage au niveau relationnel et limites....24
IV-C - Les contraintes entre associations....25
IV-C-1 - La contrainte d'inclusion....25
IV-C-2 - La contrainte d'exclusion....26
IV-C-3 - La contrainte de totalité....26
IV-C-4 - La contrainte d'égalité....27
IV-C-5 - Combinaison entre contraintes....27
IV-D - Les CIF (contraintes d'intégrités fonctionnelles) et agrégations....28
V - Conclusion....29
VI - Remerciements....29

I - MERISE au service des systèmes d'information

I-A - Le système d'information

Le **système d'information** ou SI, peut être défini comme étant l'ensemble des moyens humains, matériels et immatériels mis en œuvre afin de gérer l'information au sein d'une unité, une entreprise par exemple.

Il ne faut toutefois pas confondre un **système d'information** avec un **système informatique**. En effet, les systèmes d'information ne sont pas toujours totalement informatisés et existaient déjà avant l'arrivée des nouvelles technologies de l'information et des communications dont l'informatique fait partie intégrante.

Le SI possède 4 fonctions essentielles :

- La **saisie** ou **collecte** de l'information
- La **mémorisation** de l'information à l'aide de fichier ou de base de données
- Le **traitement** de l'information afin de mieux l'exploiter (consultation, organisation, mise à jour, calculs pour obtenir de nouvelles données, ...)
- La **diffusion** de l'information

Autrefois, l'information était stockée sur papier à l'aide de formulaires, de dossiers, … et il existait des procédures manuelles pour la traiter. Aujourd'hui, les systèmes informatisées, comme les systèmes de gestion de bases de données relationnelles (SGBDR), sont mis au service du système d'information.

I-B - MERISE

MERISE est une méthode française née dans les années 70, développée initialement par **Hubert Tardieu**. Elle fut ensuite mise en avant dans les années 80, à la demande du Ministère de l'Industrie qui souhaitait une méthode de conception des SI.

MERISE est donc une méthode d'analyse et de conception des SI basée sur le principe de la séparation des données et des traitements. Elle possède un certain nombre de **modèles** (ou **schémas**) qui sont répartis sur 3 niveaux :

- Le niveau **conceptuel**,
- Le niveau **logique** ou **organisationnel**,
- Le niveau **physique**.

Dans ce cours, nous ne nous intéresserons qu'à certains schémas permettant la conception d'une base de données relationnelle puis sa réalisation sur un SGBDR.

II - Modélisation d'une base de données au niveau conceptuel

Il s'agit de l'élaboration du **modèle conceptuel des données** (MCD) qui est une représentation graphique et structurée des informations mémorisées par un SI. Le MCD est basé sur deux notions principales : les **entités** et les **associations**, d'où sa seconde appellation : le **schéma Entité/Association**.

L'élaboration du MCD passe par les étapes suivantes :

- La mise en place de **règles de gestion** (si celles-ci ne vous sont pas données),
- L'élaboration du **dictionnaire des données**,
- La recherche des **dépendances fonctionnelles** entre ces données,
- L'élaboration du MCD (création des **entités** puis des **associations** puis ajout des **cardinalités**).

II-A - Les règles de gestion métiers

Avant de vous lancer dans la création de vos tables (ou même de vos entités et associations pour rester dans un vocabulaire conceptuel), il vous faut recueillir les besoins des futurs utilisateurs de votre application. Et à partir de ces besoins, vous devez être en mesure d'établir les règles de gestion des données à conserver.

Prenons l'exemple d'un développeur qui doit informatiser le SI d'une bibliothèque. On lui fixe les règles de gestion suivantes :

- Pour chaque livre, on doit connaître le titre, l'année de parution, un résumé et le type (roman, poésie, science fiction, ...).
- Un livre peut être rédigé par aucun (dans le cas d'une œuvre anonyme), un ou plusieurs auteurs dont on connaît le nom, le prénom, la date de naissance et le pays d'origine.
- Chaque exemplaire d'un livre est identifié par une référence composée de lettres et de chiffres et ne peut être paru que dans une et une seule édition.
- Un inscrit est identifié par un numéro et on doit mémoriser son nom, prénom, adresse, téléphone et adresse e-mail.
- Un inscrit peut faire zéro, un ou plusieurs emprunts qui concernent chacun un et un seul exemplaire. Pour chaque emprunt, on connaît la date et le délai accordé (en nombre de jours).

Ces règles vous sont parfois données mais vous pouvez être amené à les établir vous-même dans deux cas :

- Vous êtes à la fois maîtrise d'œuvre (MOE) et maîtrise d'ouvrage (MOA), et vous développez une application pour votre compte et/ou selon vos propres directives.
- **Ce qui arrive le plus souvent :** les futurs utilisateurs de votre projet n'ont pas été en mesure de vous fournir ces règles avec suffisamment de précision ; c'est pourquoi vous devrez les interroger afin d'établir vous même ces règles. N'oubliez jamais qu'en tant que développeur, vous avez un devoir d'assistance à maîtrise d'ouvrage si cela s'avère nécessaire.

II-B - Le dictionnaire des données

C'est une étape intermédiaire qui peut avoir son importance, surtout si vous êtes plusieurs à travailler sur une même base de données, d'un volume conséquent.

Le dictionnaire des données est un document qui regroupe toutes les données que vous aurez à conserver dans votre base (et qui figureront donc dans le MCD). Pour chaque donnée, il indique :

- Le **code mnémonique** : il s'agit d'un libellé désignant une donnée (par exemple «*titre_l*» pour le titre d'un livre)
- La **désignation** : il s'agit d'une mention décrivant ce à quoi la donnée correspond (par exemple «*titre du livre*»)
- Le **type de donnée** :
  - **A** ou **Alphabétique** : lorsque la donnée est uniquement composée de caractères alphabétiques (de 'A' à 'Z' et de 'a' à 'z')
  - **N** ou **Numérique** : lorsque la donnée est composée uniquement de nombres (entiers ou réels)
  - **AN** ou **Alphanumérique** : lorsque la donnée peut être composée à la fois de caractères alphabétiques et numériques
  - **Date** : lorsque la donnée est une date (au format AAAA-MM-JJ)
  - **Booléen** : Vrai ou Faux
- La **taille** : elle s'exprime en nombre de caractères ou de chiffres. Dans le cas d'une date au format AAAA-JJ-MM, on compte également le nombre de caractères, soit 10 caractères. Pour ce qui est du type booléen, nul besoin de préciser la taille (ceci dépend de l'implémentation du SGBDR).
- Et parfois des **remarques** ou **observations** complémentaires (par exemple si une donnée est strictement supérieure à 0, etc).

Reprenons l'exemple de notre bibliothèque et du système de gestion des emprunts que nous sommes chargés d'informatiser. Après l'étude des règles de gestion, nous pouvons établir le dictionnaire des données suivant :

Code mnémonique	Désignation	Type	Taille	Remarque
id_i	Identifiant numérique d'un inscrit	N		
nom_i	Nom d'un inscrit	A	30	
prenom_i	Prénom d'un inscrit	A	30	
rue_i	Rue où habite un inscrit	AN	50	
ville_i	Ville où habite un inscrit	A	50	
cp_i	Code postal d'un inscrit	AN	5	
tel_i	Numéro de téléphone fixe d'un inscrit	AN	15	
tel_port_i	Numéro de téléphone portable d'un inscrit	AN	15	
email_i	Adresse e-mail d'un inscrit	AN	100	
date_naissance_i	Date de naissance d'un inscrit	Date	10	Au format AAAA-JJ-MM
id_l	Identifiant numérique d'un livre	N		
titre_l	Titre d'un livre	AN	50	
annee_l	Année de parution d'un livre	N	4	
resume_l	Résumé d'un livre	AN	1000	
ref_e	Code de référence d'un exemplaire d'un livre	AN	15	Cette référence servira également d'identifiant dans ce système
id_t	Identifiant numérique d'un type de livre	N		
libelle_t	Libellé d'un type de livre	AN	30	
id_ed	Identifiant numérique	N	6	

	d'une édition de livre			
nom_ed	Nom d'une édition de livre	AN	30	
id_a	Identifiant numérique d'un auteur	N		
nom_a	Nom d'un auteur	A	30	
prenom_a	Prénom d'un auteur	A	30	
date_naissance	Date de naissance d'un auteur	Date		Au format AAAA-JJ-MM
id_p	Identifiant numérique d'un pays	N		
nom_p	Nom d'un pays	A	50	
id_em	Identifiant numérique d'un emprunt	N		
date_em	Date de l'emprunt	Date		Au format AAAA-JJ-MM
delais_em	Délai autorisé lors de l'emprunt du livre	N	3	S'exprime en nombre de jours

Remarques :

- Les données qui figurent dans le MCD (et donc dans le dictionnaire des données) doivent être, dans la plupart des cas, **élémentaires** :
  - Elles ne doivent pas être **calculées** : les données calculées doivent être obtenues, par le calcul, à partir de données élémentaires qui, elles, sont conservées en base. Cependant, il existe quelques cas où il s'avère pertinent de conserver, pour des raisons d'optimisation, une donnée calculée, le montant d'une commande par exemple. On ne conservera cependant pas les données calculées intermédiaires sauf en cas d'obligation légale (c'est le cas pour un montant HT par exemple, où les composantes peuvent d'ailleurs avoir un prix variable dans le temps). En effet, cela évite de refaire les calculs plusieurs fois pour un résultat qui restera fixe.
  - Elles ne doivent pas être **composées** : les données composées doivent être obtenues par la concaténation de données élémentaires conservées en base. Par exemple une adresse est obtenue à partir d'une rue, d'une ville et d'un code postal : ce sont ces trois dernières données qui sont conservées et donc qui figureront dans le MCD (et dans le dictionnaire des données).
- Lorsque l'on n'effectue jamais de calcul sur une donnée numérique, celle-ci doit être de type AN (c'est le cas par exemple pour un numéro de téléphone).

II-C - Les dépendances fonctionnelles

Soit deux propriétés (ou données) P1 et P2. On dit que P1 et P2 sont reliées par une **dépendance fonctionnelle** (DF) si et seulement si une **occurrence** (ou valeur) de P1 permet de connaître une et une seule occurrence de P2.

Cette dépendance est représentée comme ceci :

P1 → P2

On dit que P1 est la **source** de la DF et que P2 en est le **but**.

Par ailleurs, plusieurs données peuvent être source comme plusieurs données peuvent être but d'une DF. Exemples :

P1,P2 → P3
P1 → P2,P3
P1, P2 → P3,P4,P5
…

En reprenant les données du dictionnaire précédent, on peut établir les DF suivantes :

id_em → date_em, delais_em, id_i, ref_e
id_i → nom_i, prenom_i, rue_i, ville_i, cp_i, tel_i, tel_port_i, email_i, date_naissance_i
ref_e → id_l
id_l → titre_l, annee_l, resume_l, id_t, id_ed
id_t → libelle_t
id_ed → nom_ed
id_a → nom_a, prenom_a, date_naissance_a, nom_p

On peut déduire les conclusions suivantes de ces DF :

- À partir d'un numéro d'emprunt, on obtient une date d'emprunt, un délai, l'identifiant de l'inscrit ayant effectué l'emprunt, la référence de l'exemplaire emprunté.
- À partir d'une référence d'exemplaire, on obtient l'identifiant du livre correspondant.
- À partir d'un numéro de livre, on obtient son titre, son année de parution, un résumé, l'identifiant du type correspondant, son numéro d'édition.
- ...

Remarque :

Une DF doit être :

- élémentaire : C'est l'intégralité de la source qui doit déterminer le but d'une DF. Par exemple si P1 → P3 alors P1,P2 → P3 n'est pas élémentaire.
- directe : La DF ne doit pas être obtenue par transitivité. Par exemple, si P1 → P2 et P2 → P3 alors P1 → P3 a été obtenue par transitivité et n'est donc pas directe.

Conclusion :

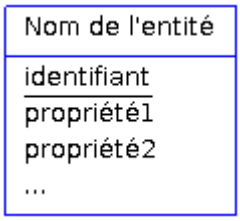
Les DF qui existent entre les données sont parfois évidentes et ne nécessitent pas toujours une modélisation mais celle-ci peut s'avérer utile car elle permet, entre autres, de distinguer les futures entités du MCD et leur identifiants.

II-D - Le Modèle Conceptuel de Données (MCD)

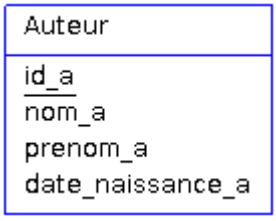
II-D-1 - Les entités

Chaque entité est unique et est décrite par un ensemble de propriétés encore appelées attributs ou caractéristiques. Une des propriétés de l'entité est l'identifiant. Cette propriété doit posséder des occurrences uniques et doit être source des dépendances fonctionnelles avec toutes les autres propriétés de l'entité. Bien souvent, on utilise une donnée de type entier qui s'incrémente pour chaque occurrence, ou encore un code unique spécifique du contexte.

Le formalisme d'une entité est le suivant :



Ainsi, si on reprend notre dictionnaire de données précédent, on schématise par exemple une entité «**Auteur**» comme ceci :



À partir de cette entité, on peut retrouver la règle de gestion suivante : un auteur est identifié par un numéro unique (id_a) et est caractérisé par un nom, un prénom et une date de naissance.

Une entité peut n'avoir aucune, une ou plusieurs occurrences. Pour illustrer ce terme d'«occurrence» qui a déjà été utilisé plusieurs fois, voici un exemple de **table d'occurrences** de l'entité **Auteur** :

id_a	nom_a	prenom_a	date_naissance_a
1	Hugo	Victor	1802-02-26
2	Rimbaud	Arthur	1854-10-20
3	de Maupassant	Guy	1850-08-05

Cette table est composée de trois occurrences de l'entité **Auteur**.

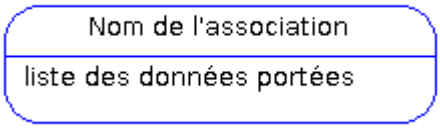
Remarques :

- Les occurrences sont parfois appelés **tuples**. Par ailleurs, la table d'occurrence peut être comparée à l'**instance** d'une **relation** (implantation relationnelle d'une entité ou association) à un moment donné. Nous reviendrons sur cette notion de relation dans la partie III.
- Au niveau conceptuel, on devrait plutôt parler d'**entités-types** , les entités étant en fait des instances d'entités-types. Par soucis de simplicité, on gardera les termes d'entités et associations tout au long du cours.

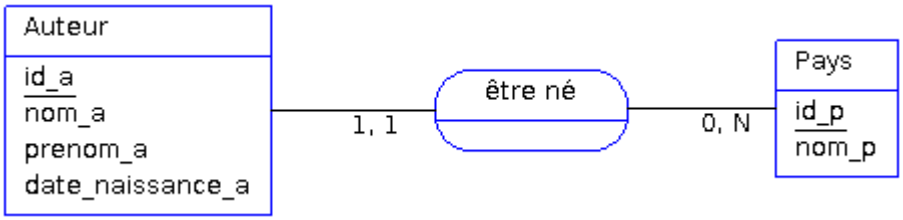
II-D-2 - Les associations

Une association définit un lien sémantique entre une ou plusieurs entités. En effet, la définition de liens entre entités permet de traduire une partie des règles de gestion qui n'ont pas été satisfaites par la simple définition des entités.

Le formalisme d'une association est le suivant :



Généralement le nom de l'association est un verbe définissant le lien entre les entités qui sont reliées par cette dernière. Par exemple :



Ici l'association «être né» traduit les deux règles de gestion suivantes :

- Un auteur est né dans **un et un seul** pays,
- Dans un pays, sont nés **aucun, un ou plusieurs** auteurs.

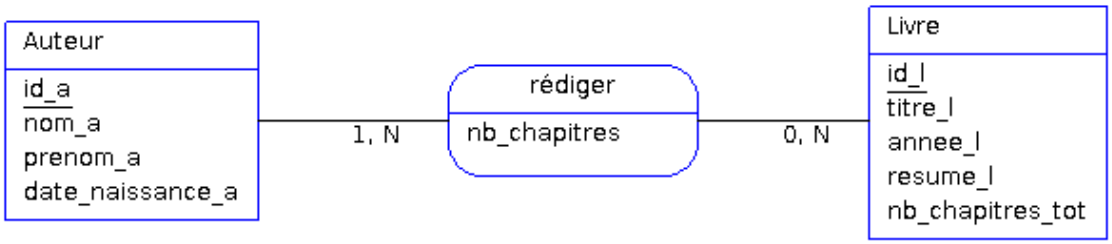
Vous remarquerez, que cette association est caractérisée par ces annotations **1,1** et **0,N** qui nous ont permis de définir les règles de gestions précédentes. Ces annotations sont appelées les **cardinalités**.

Une cardinalité est définie comme ceci :

minimum, maximum

Les cardinalités les plus répandues sont les suivantes : **0,N** ; **1,N** ; **0,1** ; **1,1**. On peut toutefois tomber sur des règles de gestion imposant des cardinalités avec des valeurs particulières, mais cela reste assez exceptionnel et la présence de ces cardinalités imposera l'implantation de traitements supplémentaires.

L'identifiant d'une association ayant des cardinalités 0,N/1,N de part et d'autre, est obtenu par la concaténation des entités qui participent à l'association. Imaginons l'association suivante :



Ici un auteur rédige au moins un ou plusieurs livres et pour chaque livre, on connaît le nombre de chapitres rédigés par l'auteur (on connaît aussi le nombre total de chapitres pour chaque livre).

L'association «rédiger» peut donc être identifiée par la concaténation des propriétés id_a et id_l. Ainsi, le couple **id_a, id_l** doit être unique pour chaque occurrence de l'association. On peut également définir la dépendance fonctionnelle suivante :

id_a, id_l → nb_chapitres

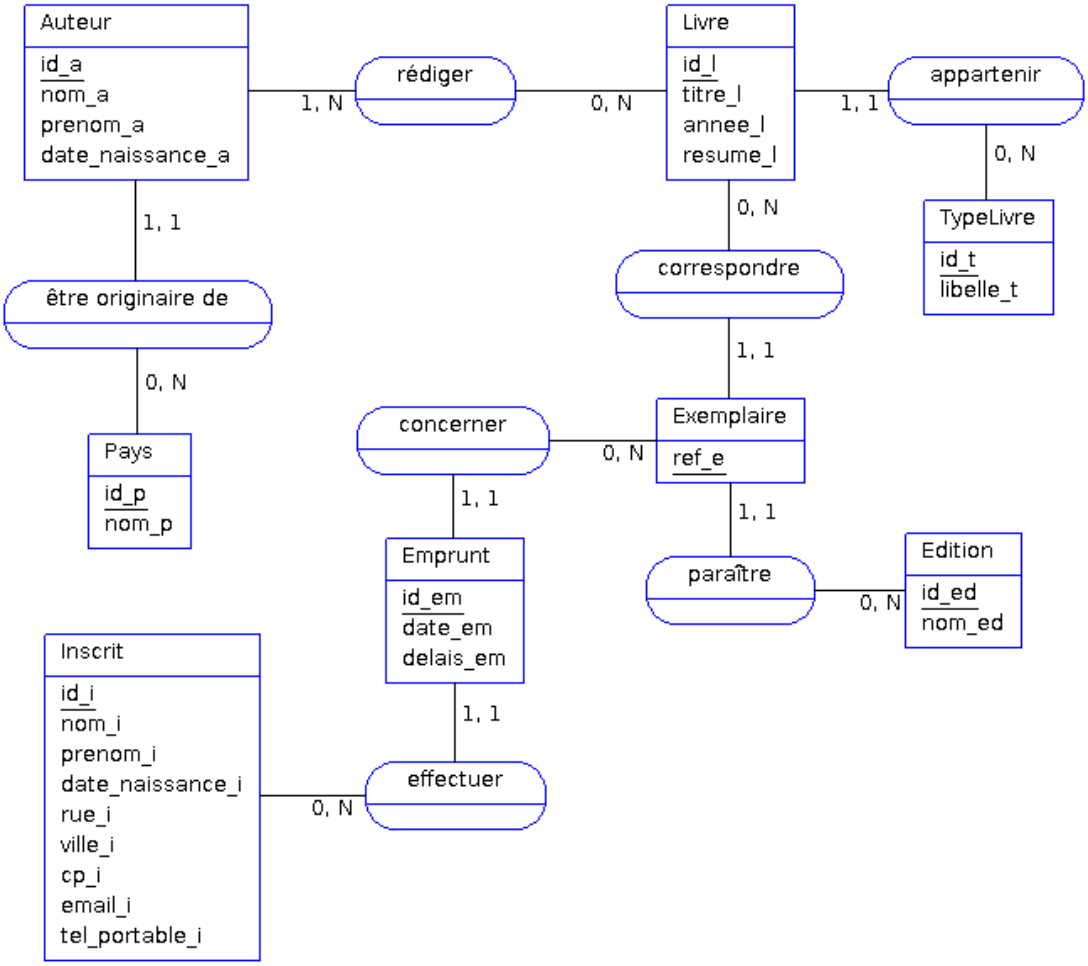
On dit que nb_chapitres (nombre de chapitres rédigés par un auteur, pour un livre) est une donnée portée par l'association «rédiger». Cette association est donc une association porteuse de données.

Pour une association ayant au moins une cardinalité de type 0,1 ou 1,1 considérons dans un premier temps que cette dernière ne peut être porteuse de données et qu'elle est identifiée par l'identifiant de l'entité porteuse de la cardinalité 0,1 ou 1,1.

Nous reviendrons plus en détail sur la notion d'identification d'une association lors du passage au modèle logique.

II-D-3 - Élaboration du MCD

Avec toutes ces connaissances, il nous est donc possible d'élaborer le MCD complet à partir des données présentes dans le dictionnaire des données :



Remarques :

- Souvent, pour un même ensemble de règles de gestion, plusieurs solutions sont possibles au niveau conceptuel. Par exemple, rien ne nous obligeait ici à créer une entité **Type**. Une simple donnée portée par l'entité **Livre** aurait pu convenir également.
- Pour que le MCD soit sémantiquement valide, toute entité doit être reliée à au moins une association.
- Les entités et les propriétés peuvent être historisées. Dans ce cas on met un **(H)** à la fin du nom de l'entité ou de la propriété que l'on souhaite historiser (cela permet de préciser que l'on archivera toutes les modifications sur une entité ou une propriété donnée). Cela doit également répondre à une règle de gestion.
- Il existe des outils de modélisation payants et d'autres gratuits pour MERISE (powerAMC, OpenModelSphere, AnalyseSI, JMerise, etc).
- On aurait pu, dans ce cas précis, conserver également une date de rentrée des livres, calculée à partir de la date de location et de la durée de celle-ci. C'est un exemple de donnée calculée dont la conservation peut s'avérer pertinente (notamment pour faciliter l'envoi de rappels).

III - Modélisation d'une base de données au niveau logique et passage au SQL

Dans cette partie, nous allons voir comment établir une modélisation des données au niveau logique (ou relationnel) à partir d'un modèle conceptuel, puis comment passer à l'étape de création des tables (cela suppose d'avoir une connaissance préalable des requêtes SQL de création de tables).

III-A - Le passage du MCD au MLD et SQL

III-A-1 - Les relations

Le modèle logique de données (MLD) est composé uniquement de ce que l'on appelle des **relations**. Ces relations sont à la fois issues des entités du MCD mais aussi d'associations, dans certains cas. Ces relations nous permettront par la suite de créer nos tables au niveau physique.

Une relation est composée d'attributs. Ces attributs sont des données élémentaires issues des propriétés des différentes entités mais aussi des identifiants et des données portées par certaines associations.

Une relation possède un nom qui correspond en général à celui de l'entité ou de l'association qui lui correspond. Elle possède aussi une **clef primaire** qui permet d'identifier sans ambiguïté chaque occurrence de cette relation. La clef primaire peut être composée d'un ou plusieurs attributs, il s'agit d'une implantation de la notion d'identifiant des entités et associations qui se répercute au niveau relationnel.

Voici un premier exemple de relation (issue de l'entité «Edition» de notre précédant MCD) :

Edition (id_ed, nom_ed)
Légende :
x : relation
x : clef primaire

Remarques :

- Ce premier MLD est représenté de manière textuelle. C'est notamment cette représentation que l'on retrouve dans beaucoup de formations d'études supérieures. Il existe toutefois une représentation graphique équivalente.
- Il est important d'accompagner un MLD textuel d'une légende (ce dernier n'ayant pas de formalisme normé). Ceci est d'ailleurs exigé dans certaines formations.

Il existe un autre type de clef appelé **clef étrangère**. La clef étrangère est un attribut d'une relation qui fait référence à la clef primaire d'une autre relation (ces deux clefs devront donc avoir le même type de données).

Complétons notre premier exemple avec une autre relation où apparaît une clef étrangère :

Edition (id_ed, nom_ed)
Exemplaire (ref_e, *id_ed#*)
Légende :
x : relation
x : clef primaire
x# : clef étrangère

Remarques :

- Au niveau relationnel, on devrait plutôt parler de **clef candidate** qui permet d'identifier sans ambiguïté une occurrence de la relation pour les clefs primaires. De même, on devrait désigner une clef étrangère par

une **contrainte d'inclusion** vers une clef candidate. Par souci de simplicité, on gardera les termes de clefs primaires et étrangères.
- Par convention, on fait précéder ou suivre la clef étrangère du symbole #. Ceci n'est pas une obligation à partir du moment où les légendes sont suffisamment précises.
- Ici la clef étrangère présente dans la relation «Exemplaire» fait référence à la clef primaire de la relation «Edition».
- Une relation peut posséder aucune, une ou plusieurs clefs étrangères mais possède toujours une et une seule clef primaire.

Enfin, vous pouvez également rencontrer le terme de **cardinalité de la relation** qui signifie ici le nombre d'occurrences d'une relation (ou nombre d'entrées dans la table correspondante) et le terme de **degré de la relation** qui correspond au nombre d'attributs d'une relation.

III-A-2 - Règles de conversion

Comme cela a déjà été dit précédemment, les relations du MLD sont issues des entités du MCD et de certaines associations. Nous allons maintenant aborder ces règles de conversion de façon plus précise.

III-A-2-a - Règle 1 - conversion d'une entité

En règle générale, toute entité du MCD devient une relation dont la clef est l'identifiant de cette entité. Chaque propriété de l'entité devient un attribut de la relation correspondante.

Il existe toutefois quelques cas particuliers que vous pourrez voir au paragraphe .

III-A-2-b - Règle 2 - conversion d'associations n'ayant que des cardinalités de type 0/1,N

Une association ayant des cardinalités 0,N ou 1,N de part et d'autre devient une relation dont la clef est constituée des identifiants des entités reliées par cette association. Ces identifiants seront donc également des clefs étrangères respectives. On parle de **relations associatives**.

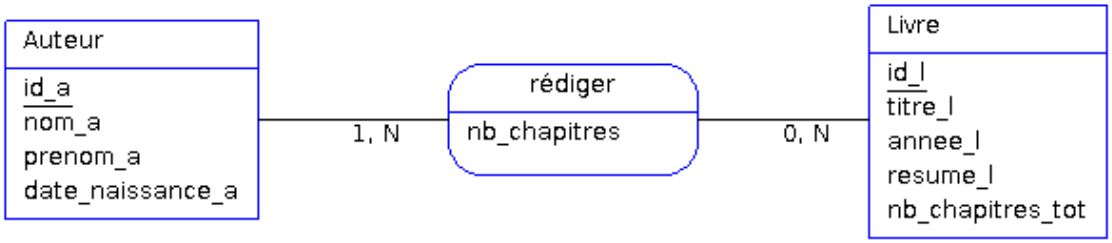
Les cardinalités plus restrictives (comme 2,3 ; 1,7 ; ...) seront perçues comme des cardinalités de type 0/1,N également (il s'agit en effet de sous-ensembles). Cependant, les règles de gestions qui ne seront plus satisfaites par cette modélisation logique devront l'être par des traitements supplémentaires (via le code de l'application qui exploite la base de donnée ou encore par des triggers (déclencheurs) si le SGBDR est suffisamment robuste).

Voici un exemple de relation associative issu de l'association «rédiger» de notre MCD :

Rediger (*id_a#*, *id_l#*)

Légende :
x : relation
x : clef primaire
x# : clef étrangère

Dans le cas d'associations porteuses de données, les données portées deviennent des attributs de la relation correspondante. Si l'on reprend cet exemple :



L'association «rédiger» devrait maintenant être traduite comme ceci :

Rediger (*id_a#*, *id_l#*, nb_chapitres)
Légende :
x : relation
x : clef primaire
x# : clef étrangère

III-A-2-c - Règle 3 - conversion des associations ayant au moins une cardinalité de type 1,1

Plusieurs possibilités s'offrent à nous pour ce cas de figure. La règle de conversion la plus répandue aujourd'hui est d'ajouter une clef étrangère dans la relation qui correspond à l'entité se situant du côté de cette cardinalité 1,1. Cette clef étrangère fera donc référence à la clef de la relation correspondant à la seconde entité reliée par l'association.

Prenons un exemple issu de l'association «être originaire de» et des entités «Auteur» et «Pays» :

Pays (nom_p)
Auteur (id_a, nom_a, prenom_a, date_naissance_a, *nom_p#*)
Légende :
x : relation
x : clef primaire
x# : clef étrangère

Lorsque l'on applique cette règle de conversion, deux restrictions s'imposent :

- L'association ne peut être porteuse de données. Les données portées sont en dépendances fonctionnelles directes avec l'identifiant de l'entité dont la clef correspondante sera référencée par une clef étrangère dans une autre relation.
- L'association doit être binaire (c'est à dire relier uniquement deux entités et pas plus).

Lorsque deux entités sont toutes deux reliées avec une cardinalité 1,1 par une même association, on peut placer la clef étrangère de n'importe quel côté. Par convention, on choisit de la placer du côté de la relation correspondant à l'entité ayant le plus de liaisons avec les autres. Certains considèrent d'ailleurs que deux entités étant reliées par une association ayant une cardinalité 1,1 des deux côtés, doivent obligatoirement fusionner. Cette règle s'appuie encore une fois sur la notion de dépendances fonctionnelles directes mais n'est pas toujours respectée (il est parfois sémantiquement préférable de garder une distinction entre les deux entités).

Une autre solution (moins répandue) consiste à créer une relation associative dont la clef est cette fois composée uniquement de la clef étrangère qui fait référence à l'identifiant de l'entité du côté opposé à la cardinalité 1,1.

Si on reprend le même exemple, voici ce que l'on devrait obtenir :

Pays (nom_p)
Auteur (id_a, nom_a, prenom_a, date_naissance_a)
EtreOriginaireDe (*id_a#*, *nom_p#*)

Légende :
x : relation
x : clef primaire
x# : clef étrangère

Dans ce cas, l'association peut être porteuse de données. Ces dernières deviendront donc des attributs de la relation associative comme dans le cas des cardinalités 0,1/N.

Il va sans dire que la première solution est aujourd'hui préférable à cette dernière en terme d'optimisation et de simplification des requêtes.

III-A-2-d - Règle 4 - conversion des associations ayant au moins une cardinalité de type 0,1 (et dont les autres cardinalités sont de type 0,1/N)

De même que pour les cardinalités 1,1, une association ayant une cardinalité 0,1 doit être binaire, et les deux mêmes possibilités s'offrent à nous :

- Créer la clef étrangère dans la relation correspondant à l'entité du côté de la cardinalité 0,1. Rappelons que dans ce cas, l'association ne peut pas être porteuse de données.
- Créer une relation associative qui serait identifié de la même façon que pour une cardinalité 1,1.

Cependant, dans le cadre d'une cardinalité 0,1, nous verrons qu'il n'est pas toujours préférable de privilégier la première méthode comme c'est le cas pour une cardinalité 1,1.

Imaginons par exemple qu'un livre puisse appartenir à 0 ou 1 catégories, on obtient le MCD suivant :

Certains diront que toutes les associations binaires de type père-fils ayant des cardinalités 1,N/0,N - 1,1/0,1 sont caractérisées par l'existence d'une dépendance fonctionnelle entre l'identifiant de l'entité père (ici id_cat) et de l'entité fils (ici id_l). Cette dépendance fonctionnelle se schématiserait ainsi :

id_l → id_cat

Dans ce cas, il apparaît logique de traduire le MCD de cette façon (première méthode) :

Categorie (id_cat, libelle_cat)
Livre (id_l, titre_l, annee_l, resume_l, *id_cat#*)
Légende :
x : relation
x : clef primaire
x# : clef étrangère
Cependant même si les SGBD le permettent (avec la valeur NULL par défaut), il n'est normalement pas permis d'avoir une clef étrangère sans valeur pour laquelle on retrouverait l'occurrence dans la relation sur laquelle on fait référence.

C'est pourquoi d'autres pensent (avec raison) qu'il vaut mieux créer une relation associative de cette manière (seconde méthode) :

Categorie (id_cat, libelle_cat)
Livre (id_l, titre_l, annee_l, resume_l)
Appartenir (*id_l#*, *id_cat#*)
Légende :
x : relation
x : clef primaire
x# : clef étrangère

La pertinence de l'une ou l'autre méthode varie en fonction du nombre d'occurrences caractérisées par la cardinalité 0 ou la cardinalité 1. En effet, lorsque les occurrences avec la cardinalité 1 sont plus nombreuses que les occurrences avec la cardinalité 0, la première méthode est préférable. Dans le cas contraire, c'est la seconde méthode qui est la plus adaptée.

Enfin, dans le cas où une association binaire possède à la fois une cardinalité 0,1 et une cardinalité 1,1 (ce qui est rarement le cas), il est préférable que la clef étrangère soit du côté de la relation correspondant à l'entité situé du côté de la cardinalité 1,1.

III-A-3 - Élaboration du MLD et passage au SQL

Avec ces différentes règles de conversion, il nous est déjà possible de convertir notre MCD au complet :

Pays (id_p, nom_p)
Auteur (id_a, nom_a, prenom_a, date_naissance_a, *id_p#*)
TypeLivre (id_t, libelle_t)
Livre (id_l, titre_l, annee_l, resume_l, *id_t#*)
Rediger (*id_a#*, *id_l#*)
Edition (id_ed, nom_ed)
Exemplaire (ref_e, *id_ed#*, *id_l#*)
Inscrit (id_i, nom_i, prenom_i, date_naissance_i, rue_i, ville_i, cp_i, email_i, tel_i, tel_portable_i)
Emprunt (id_em, date_em, delais_em, *id_i#*, *ref_e#*)
Légende :
x : relation
x : clef primaire
x# : clef étrangère

Comme vous pouvez le constater, le schéma de la base est déjà fait. Les règles de passage au SQL sont assez simples :

- chaque relation devient une table
- chaque attribut de la relation devient une colonne de la table correspondante
- chaque clef primaire devient une **PRIMARY KEY**
- chaque clef étrangère devient une **FOREIGN KEY**

Voici ce que cela donnerait :

```
CREATE TABLE Pays (
    id_p INT NOT NULL,
    nom_p VARCHAR(50),
    PRIMARY KEY (id_p)
);

CREATE TABLE Auteur (
    id_a INT NOT NULL,
    nom_a VARCHAR (30),
    prenom_a VARCHAR (30),
    date_naissance_a DATE,
    id_p INT NOT NULL,
    FOREIGN KEY (id_p) REFERENCES Pays(id_p),
```

```
    PRIMARY KEY (id_a)
);

CREATE TABLE TypeLivre (
    id_t INT NOT NULL,
    libelle_t VARCHAR (30),
    PRIMARY KEY (id_t)
);

CREATE TABLE Livre (
    id_l INT NOT NULL,
    titre_l VARCHAR (254),
    annee_l VARCHAR (4),
    resume_l TEXT,
    id_t INT NOT NULL,
    FOREIGN KEY (id_t) REFERENCES TypeLivre(id_t),
    PRIMARY KEY (id_l)
);

CREATE TABLE Rediger (
    id_a INT NOT NULL,
    id_l INT NOT NULL,
    FOREIGN KEY (id_a) REFERENCES Auteur(id_a),
    FOREIGN KEY (id_l) REFERENCES Livre (id_l),
    PRIMARY KEY (id_a, id_l)
);

CREATE TABLE Edition (
    id_ed INT NOT NULL,
    nom_ed VARCHAR (254),
    PRIMARY KEY (id_ed)
);

CREATE TABLE Exemplaire (
    ref_e VARCHAR(254) NOT NULL,
    id_ed INT NOT NULL,
    id_l INT NOT NULL,
    FOREIGN KEY (id_ed) REFERENCES Edition (id_ed),
    FOREIGN KEY (id_l) REFERENCES Livre(id_l),
    PRIMARY KEY (ref_e)
);

CREATE TABLE Inscrit (
    id_i INT NOT NULL,
    nom_i VARCHAR (30),
    prenom_i VARCHAR (30),
    date_naissance_i DATE,
    rue_i VARCHAR(50),
    ville_i VARCHAR(50),
    cp_i VARCHAR (5),
    tel_i VARCHAR(15),
    tel_portable_i VARCHAR(15) ,
    email_i VARCHAR(100),
    PRIMARY KEY (id_i)
);

CREATE TABLE Emprunt (
    id_em INT NOT NULL,
    date_em DATE,
    delais_em INT DEFAULT 0,
    id_i INT NOT NULL,
    ref_e VARCHAR (254) NOT NULL,
    FOREIGN KEY (id_i) REFERENCES Inscrit(id_i),
    FOREIGN KEY (ref_e) REFERENCES Exemplaire(ref_e),
    PRIMARY KEY (id_em)
);
```

Remarque : Il est possible de ne pas avoir à gérer l'incrémentation des identifiants par soi-même lors des INSERT avec la plupart des SGBD.

Exemple d'auto-incrémentation sous MySQL :

```
CREATE TABLE TypeLivre (
    id_t INT AUTO_INCREMENT,
    libelle_t VARCHAR (30),
    PRIMARY KEY (id_t)
);
```

Sous PostgreSQL :

```
CREATE TABLE TypeLivre (
    id_t SERIAL,
    libelle_t VARCHAR (30),
    PRIMARY KEY (id_t)
);
```

SERIAL créera implicitement une séquence qui s'incrémente avec un pas de 1. Sous Oracle, il faudrait créer soi-même cette séquence.

III-B - Règles de vérification des niveaux de normalisation

Il existe différents niveaux de normalisation (ou formes normales). Les 3 premiers niveaux de normalisations sont les plus répandus et les plus appliqués.

La classification de ces trois premiers niveaux de normalisation repose sur les dépendances fonctionnelles entre la clef primaire de la relation et ses autres attributs.

Pour être en première forme normale (1FN ou 1NF) : Les attributs d'une relation doivent être atomiques et doivent être en dépendance fonctionnelle avec la clef primaire de cette dernière.

Pour être en deuxième forme normale (2FN ou 2NF) : Il faut être en 1FN et que toutes les dépendances fonctionnelles entre la clef primaire et les autres attributs de la relation soient élémentaires. Autrement dit, les attributs doivent dépendre de la totalité de la clef.

Pour être en troisième forme normale (3FN ou 3NF) : Il faut être en 2FN et que toutes les dépendances fonctionnelles entre la clef primaire de la relation et les autres attributs soient directes.

Il ne s'agit pas de définitions précises mais de simples règles de vérification des trois premiers niveaux de normalisation.
Pour plus de détails sur les formes normales, vous pouvez consulter ***ce cours****.*

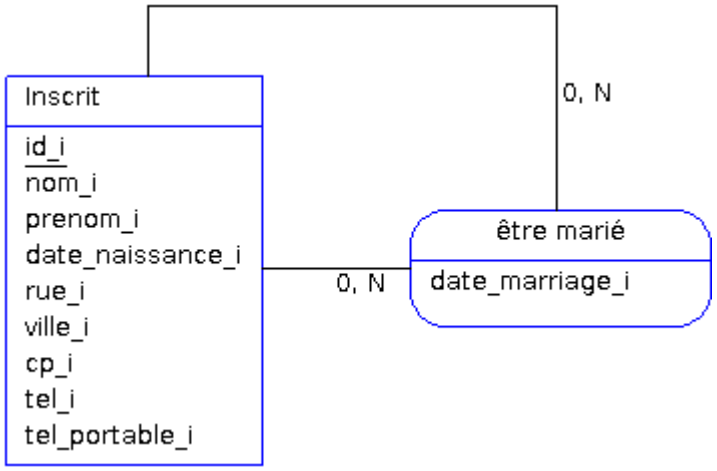
Remarques :

- Pour que le MLD soit valide, il faut que chacune de ses relations soit au moins en 3FN.
- Si un MCD est correctement conçu et que les règles de conversion énoncées plus haut ont bien été respectées, les relations seront donc automatiquement normalisées en 3FN.

III-C - Cas particuliers

III-C-1 - Les associations réflexives

Il est possible de relier une entité à elle même par une association, on parle dans ce cas là d'**association réflexive.** Imaginons que l'on veuille connaître les inscrits qui sont mariés entre eux tout en conservant leur date de mariage, voici ce que l'on obtiendrait au niveau conceptuel :



Dans ce cas, c'est la même . Il faudra cependant différencier les noms des clefs étrangères de la table associative correspondantes tout en référençant la même clef primaire :

Inscrit (id_i, nom_i, prenom_i, date_naissance_i, rue_i, ville_i, cp_i, email_i, tel_i, tel_portable_i)
EtreMarie (*id_epoux#*, *id_epouse#*, date_mariage_i)
Légende :
x : relation
x : clef primaire
x# : clef étrangère

On aurait pu choisir des cardinalités 1,1 et mettre la date de mariage comme donnée de l'entité Inscrit. Ce modèle permet tout de même de mettre la date de mariage en commun avec deux inscrits (ce qui est plus juste au niveau des dépendances fonctionnelles). Si l'on souhaite limiter le nombre de mariages à 1 pour une personne, il suffira de mettre un place un traitement qui vérifiera le nombre d'occurrence pour un inscrit dans la relation EtreMarie.

Comme exemple de traitement de vérification, nous pouvons utiliser un trigger si le SGBDR le permet. Voici un exemple de trigger avec une procédure stockée vérifiant la présence d'une occurrence pour un identifiant donnée dans la table **EtreMarie** :

```
/* Procédure stockée se chargeant de la vérification */
CREATE OR REPLACE FUNCTION verif_mariage () RETURNS TRIGGER AS $$
    DECLARE nb INT;
    BEGIN
        SELECT INTO nb COUNT(*) FROM EtreMarie WHERE id_epoux = NEW.epoux OR
 id_epouse = NEW.id_epouse;
        IF (nb >= 1) THEN
            /* Le RAISE EXCEPTION bloquera la suite des traitements dont L'INSERT */
            RAISE EXCEPTION 'Mariage impossible !';
        END IF;
        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

/* Trigger qui se déclenchera avant chaque INSERT dans la table EtreMarie */
CREATE TRIGGER t_verif_mariage BEFORE INSERT
    ON auteur FOR EACH ROW
    EXECUTE PROCEDURE verif_mariage();
```

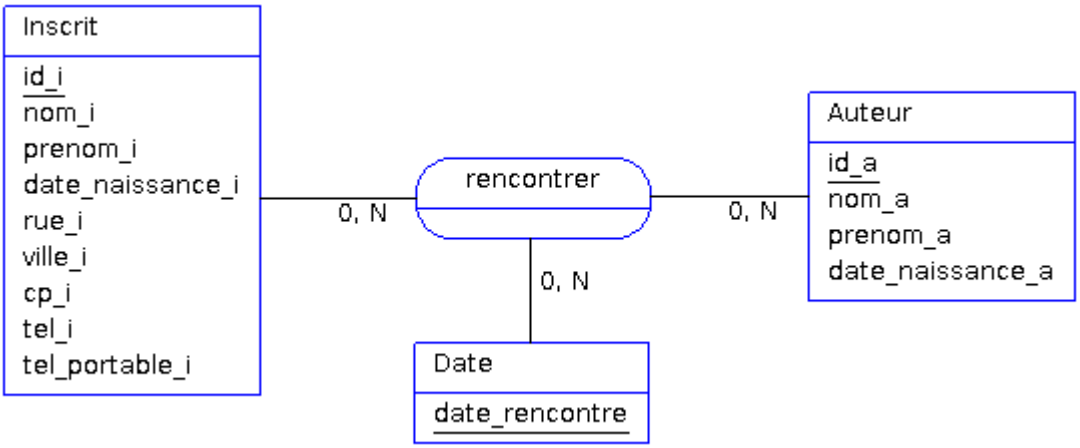
Pour cet exemple, nous avons choisi le langage PL/PgSQL qui est propre au SGBDR PostgreSQL. Vous pourrez toutefois trouver des syntaxes ressemblantes ou équivalentes sur une grande partie des SGBDR connus. C'est ce type de traitements qui permettent de répondre aux règles de gestion non satisfaites par le MCD.

Ne pas limiter le nombre d'occurrences de cette relation, permettrai en outre de conserver les différents mariages des inscrits en cas de divorce (l'intérêt est certes très limité dans le contexte de la gestion des emprunts pour une bibliothèque).

III-C-2 - Règle de conversion exceptionnelle pour certaines entités simples

Dans certains cas, il n'est pas toujours pertinent de convertir une entité au niveau conceptuel, par une relation au niveau logique. C'est le cas pour certaines entités simplement composées d'un identifiant, à l'exemple des entités de type **Date** ou **Heure** qui sont souvent utilisées dans des associations ternaires.

Imaginons par exemple que des inscrits auraient le privilège de rencontrer un auteur à une date donnée (une rencontre organisée par la bibliothèque). La rencontre est organisée avec un nombre de places limité, il faut donc garder une trace de ceux qui ont déjà fait une rencontre afin de favoriser ceux qui n'ont pas encore eu cette chance. Voici comment nous pourrions représenter cela au niveau conceptuel :



La date de rencontre ne doit pas être une simple donnée portée par l'association car cela limiterait le nombre de rencontre d'un inscrit avec un auteur à 1 (la relation correspondant à l'association aurait dans ce cas un couple identifiant unique qui imposerait cette restriction).

Le fait de créer une relation **Date** aurait pour incidence de créer de la redondance inutile, c'est pourquoi, il est recommandé dans ce cas de figure, de passer au niveau logique de cette façon :

Inscrit (id_i, nom_i, prenom_i, date_naissance_i, rue_i, ville_i, cp_i, email_i, tel_i, tel_portable_i)
Auteur (id_a, nom_a, prenom_a, date_naissance_a, *nom_p#*)
Rencontrer (*id_a#*, *id_i#*, date_rencontre)
Légende :
x : relation
x : clef primaire
x# : clef étrangère

IV - Les extensions apportées par MERISE II

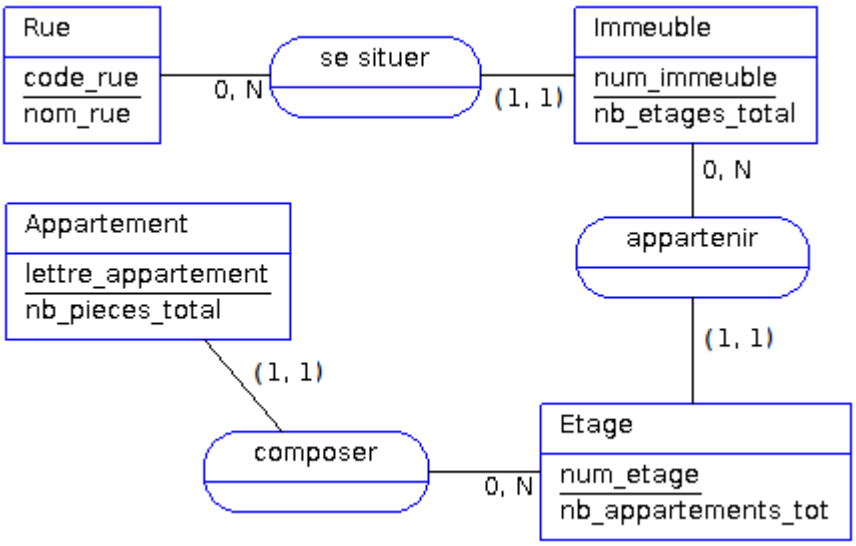
IV-A - L'identification relative

Elle intervient lorsque l'identifiant d'une entité ne suffit pas à l'identifier de manière unique.

Quelques exemples de cas où cela peut arriver :

- On identifie un immeuble par son numéro de rue, or il faut connaître le nom ou l'identifiant de la rue pour trouver l'immeuble (14 rue du général Leclerc, ...).
- On identifie un appartement par une lettre mais il faut connaître le numéro d'étage pour le retrouver (appartement A au premier étage, ...).
- Pour identifier un étage, il faut connaître l'immeuble dans lequel il est situé.

Voici comment on pourrait schématiser ces règles de gestions au niveau conceptuel :



Les parenthèses autour des cardinalités signifient que les entités du côté de ces cardinalités seront identifiées par la concaténation de leurs identifiants (qui ne suffisent pas à les identifier de manière unique) avec l'identifiant de l'entité opposée. Ainsi on obtient au niveau relationnel :

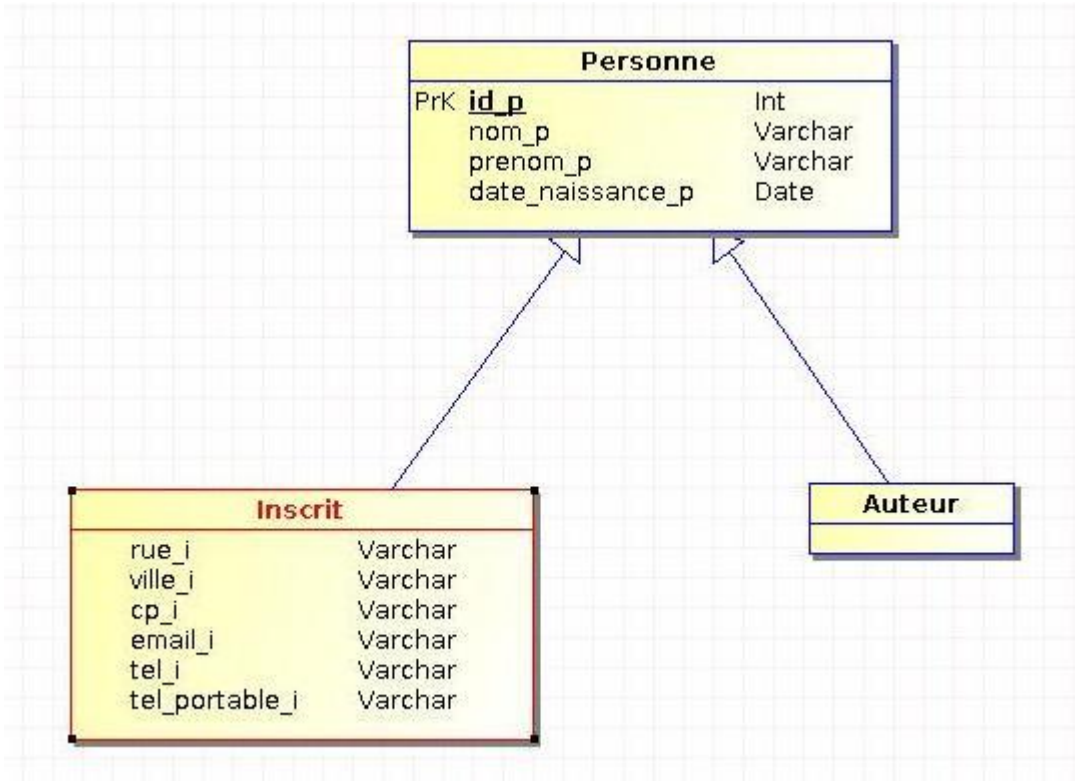
Rue (code_rue, nom_rue)
Immeuble (num_immeuble, *code_rue#*, nb_etages_total)
Etage (num_etage, *num_immeuble#*, *code_rue#*, nb_appartements_tot)
Appartement (lettre_appartement, *num_etage#*, *num_immeuble#*, *code_rue#*, nb_pieces_total)

Légende :
x : relation
x : clef primaire
x# : clef étrangère

IV-B - L'héritage et ses limites

Désormais, MERISE II permet aussi de modéliser l'héritage entre les entités. L'héritage a du sens lorsque plusieurs entités possèdent des propriétés similaires. On parle alors de généralisation avec un sur-type (ou entité mère) et de spécialisation avec des sous-type (entités filles).

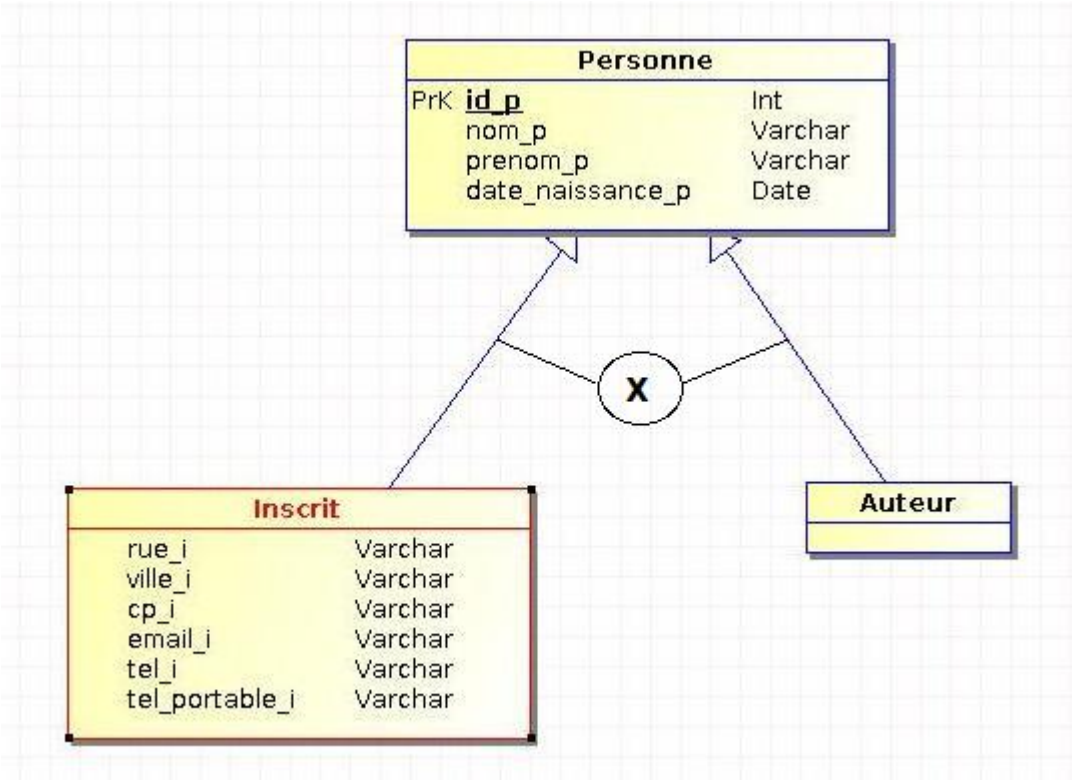
Voici comment on pourrait représenter un héritage sur notre MCD :



Dans cette partie, les types des propriétés apparaissent (ceci est dû au logiciel utilisé qui est plus adapté pour représenter l'héritage). Cependant, les types ne devraient pas être représentés au niveau conceptuel.

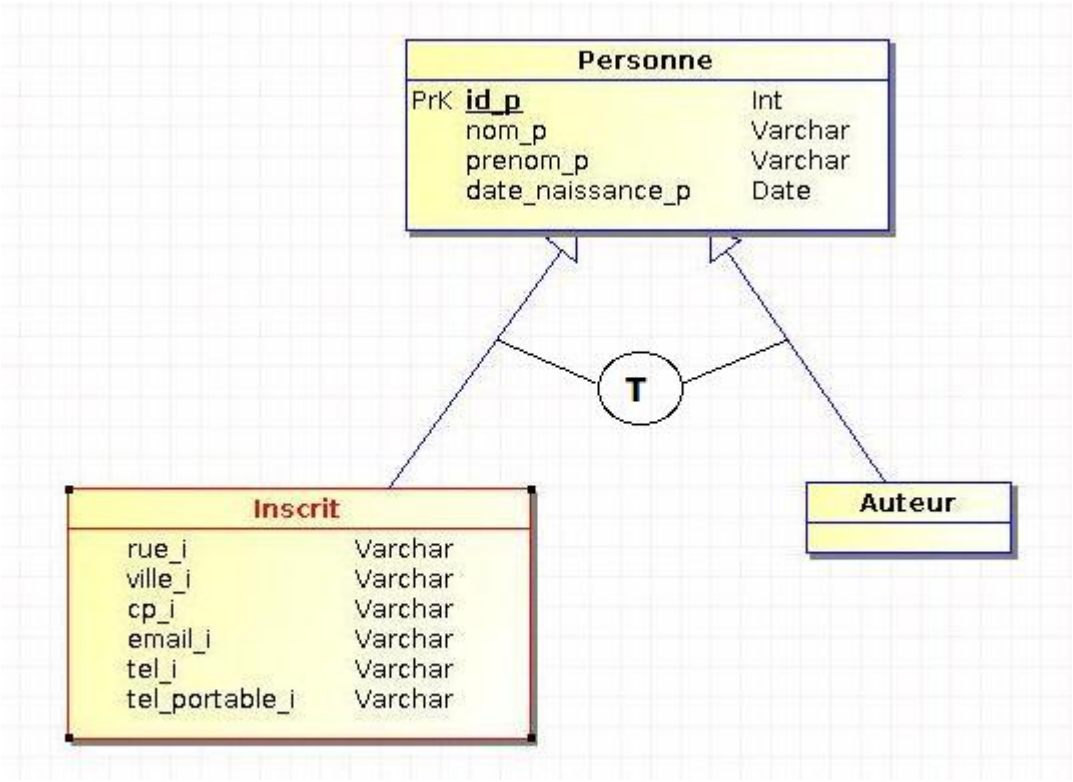
Il existe différents types d'héritage : l'héritage sans contraintes comme nous venons de le représenter, l'héritage par disjonction (ou exclusion), l'héritage par couverture (ou totalité) et enfin l'héritage par partition (totalité et exclusion).

IV-B-1 - L'héritage par disjonction (ou exclusion)



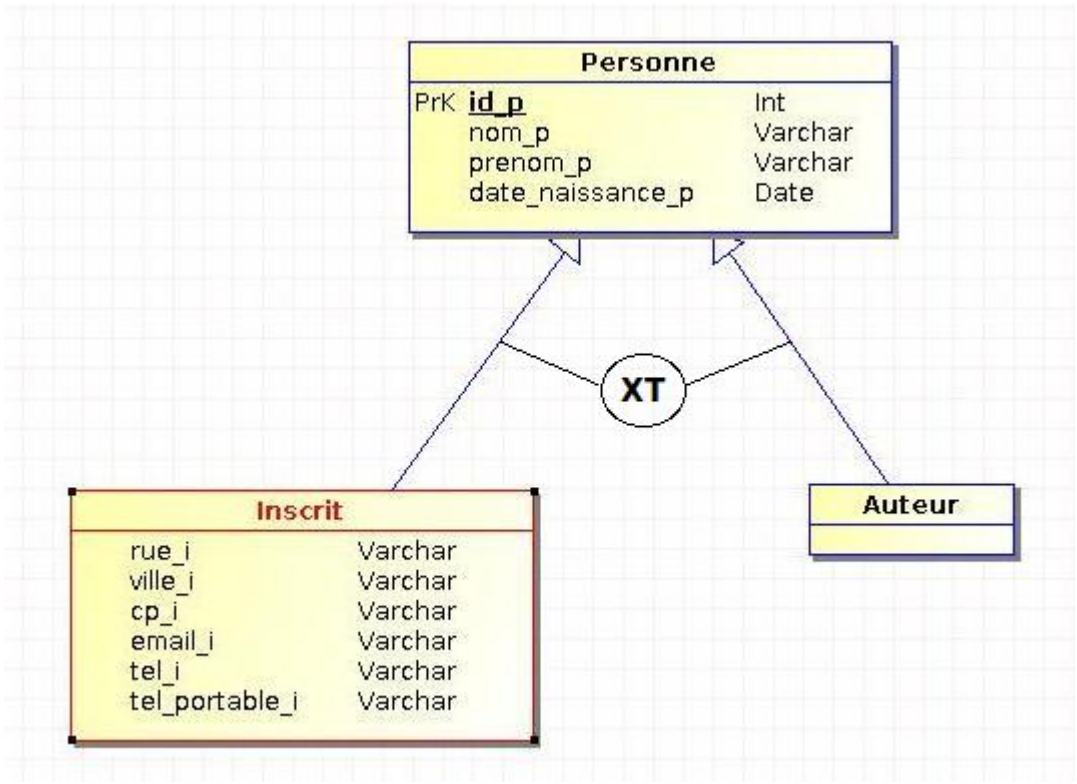
Toutes les occurrences du sur-type ne peuvent se trouver que dans aucun ou un seul sous-type. Dans notre exemple ci-dessus, un auteur ne peut pas être également un inscrit et un inscrit ne peut pas être également un auteur (une personne peut être un auteur, un inscrit ou quelqu'un d'autre).

IV-B-2 - L'héritage par couverture (ou totalité)



Toutes les occurrences du sur-type se trouvent dans au moins un des sous-type existants. Dans notre exemple, une personne est forcément un auteur ou un inscrit (ou les deux).

IV-B-3 - L'héritage par partition (totalité et exclusion)



Il s'agit d'une combinaison des deux héritages précédents : toutes les occurrences du sur-type se trouvent forcement dans un et un seul des sous-types. Une personne est soit un auteur, soit un inscrit. Cette contrainte est parfois notée «+».

IV-B-4 - Passage au niveau relationnel et limites

À son apparition avec Merise II, l'héritage n'était pas encore implanté sur l'ensemble des SGBDR répandus (ce n'est d'ailleurs toujours pas le cas aujourd'hui). Il a donc fallu le simuler au point de vue relationnel.

De façon générale, l'héritage peut être implanté au niveau relationnel en utilisant une clef étrangère vers la relation mère, comme clef primaire pour les relations filles. Reprenons notre exemple précédent :

Personne (id_p, nom_p, prenom_p, date_naissance_p)
Inscrit (*id_p#*, rue_i, ville_i, cp_i, email_i, tel_i, tel_portable_i)
Auteur (*id_p#*)
Légende :
x : relation
x : clef primaire
x# : clef étrangère

Ainsi pour satisfaire les contraintes de totalité, d'exclusion ou de partition il faudra mettre en place des traitements supplémentaires au niveau de la base de préférence (triggers, procédures stockées).

Aujourd'hui, la plupart des SGBDR performants sont capables de gérer eux-mêmes l'héritage. C'est notamment le cas avec la clause INHERITS de PostgreSQL. Cette solution est en général préférable parce qu'elle évite les jointures coûteuses entre tables mères et filles. En effet les attributs du sur-type seront automatiquement accessibles depuis le sous-type. Par conséquent, une insertion, modification ou suppression dans le sous-type se répercutera également

dans l'entité mère. Cependant, cette solution a pour inconvénient de pouvoir créer des tuples en double au niveau de la relation mère et de violer ainsi les contraintes d'intégrités référentielles en quelque sorte.

Par ailleurs, certains font parfois abstraction de la relation mère dans le cas d'un héritage par partition, et se contentent de créer les relations filles comme relations distinctes ne partageant pas de données communes. Exemple :

Auteur (id_a, nom_a, prenom_a, date_naissance_a)
Inscrit (id_i, nom_i, prenom_i, date_naissance_i, rue_i, ville_i, cp_i, email_i, tel_i, tel_portable_i)

Légende :
x : relation
x : clef primaire
x# : clef étrangère
Cette solution est également acceptable mais peut nous amener à nous interroger sur la pertinence de l'héritage étant donné que ce dernier n'est pas implanté au niveau relationnel. Cependant, la contrainte de partition reste une règle de gestion à satisfaire d'où l'importance de la modélisation de celle-ci au niveau conceptuel.

Pour conclure, bien qu'appréciée par l'enseignement, la notion d'héritage est très souvent mise de côté par les développeurs dans le cadre d'une base de donnée relationnelle.

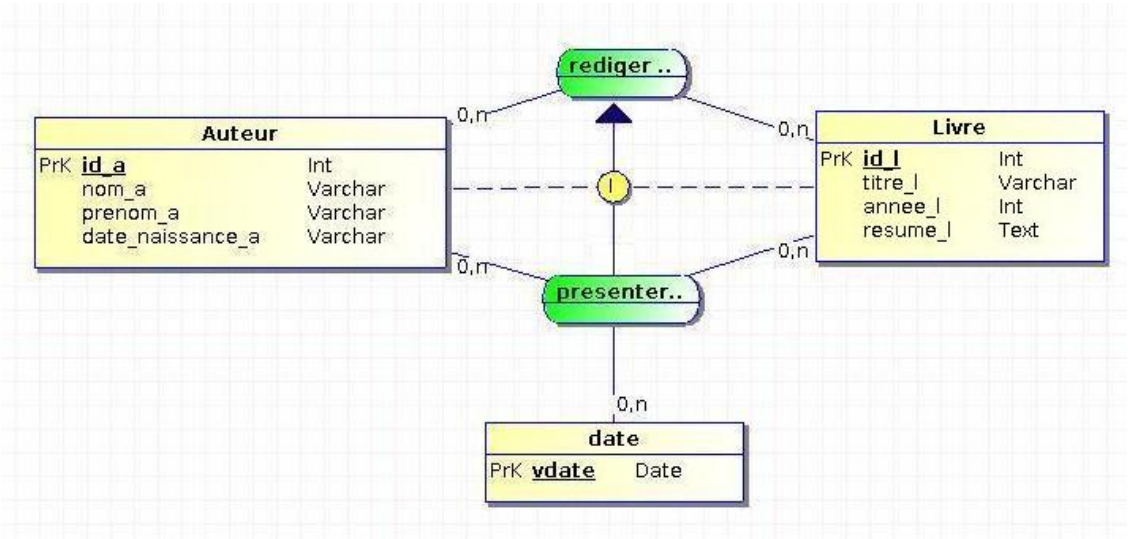
IV-C - Les contraintes entre associations

De même que pour l'héritage, il existe différentes contraintes qui peuvent exister entre deux ou plusieurs associations. Bien que non implantées au niveau relationnel, ces contraintes qui sont des règles de gestion devront être satisfaites par des traitements supplémentaires (triggers, etc).

IV-C-1 - La contrainte d'inclusion

La présence d'occurrences d'une ou plusieurs associations doit obligatoirement se répercuter sur l'association cible de la contrainte d'inclusion.

Par exemple imaginons qu'on l'on souhaite recenser les présentations des ouvrages par leur auteurs à une date donnée. Voici comment cela pourrait être représenté (de manière simpliste) au niveau conceptuel :

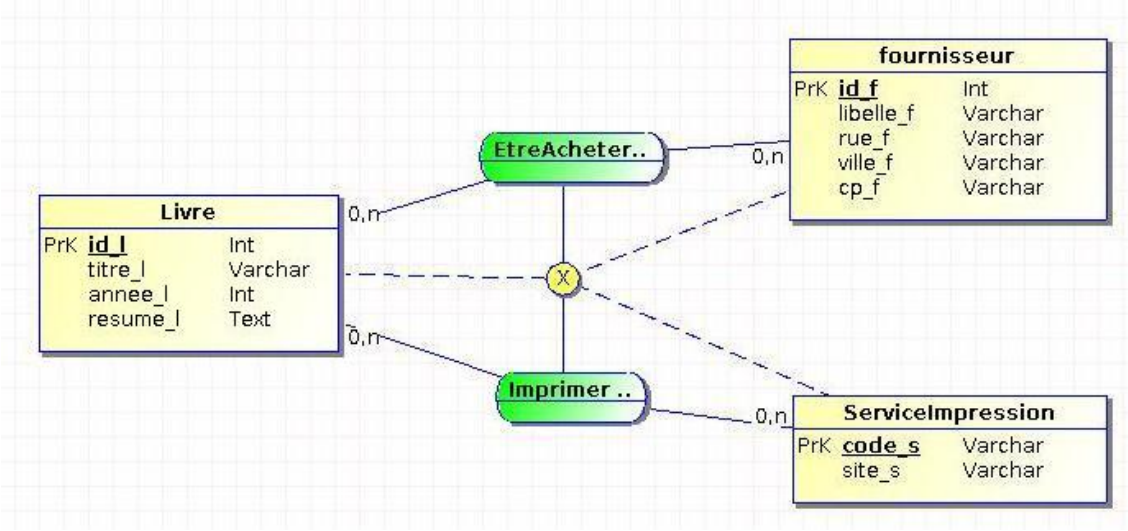


Cela signifie que si un couple livre-auteur est présent dans l'association «presenter», alors il doit obligatoirement être présent dans l'association «rediger». Cela traduit simplement la règle de gestion qui impose que pour qu'un auteur fasse la promotion d'un ouvrage, il doit en être l'un des écrivains.

IV-C-2 - La contrainte d'exclusion

Lorsqu'une occurrence est présente dans l'une des associations concernées par la contrainte d'exclusion, elle ne doit pas être présente dans une des autres associations concernées par cette contrainte.

Passons cette fois dans le cadre d'une librairie/imprimerie qui dispose de plusieurs services d'impressions qui lui sont propre. On souhaite déterminer quels sont les livres imprimés et les livres achetés tout en gardant une trace des fournisseurs et des services d'impressions :

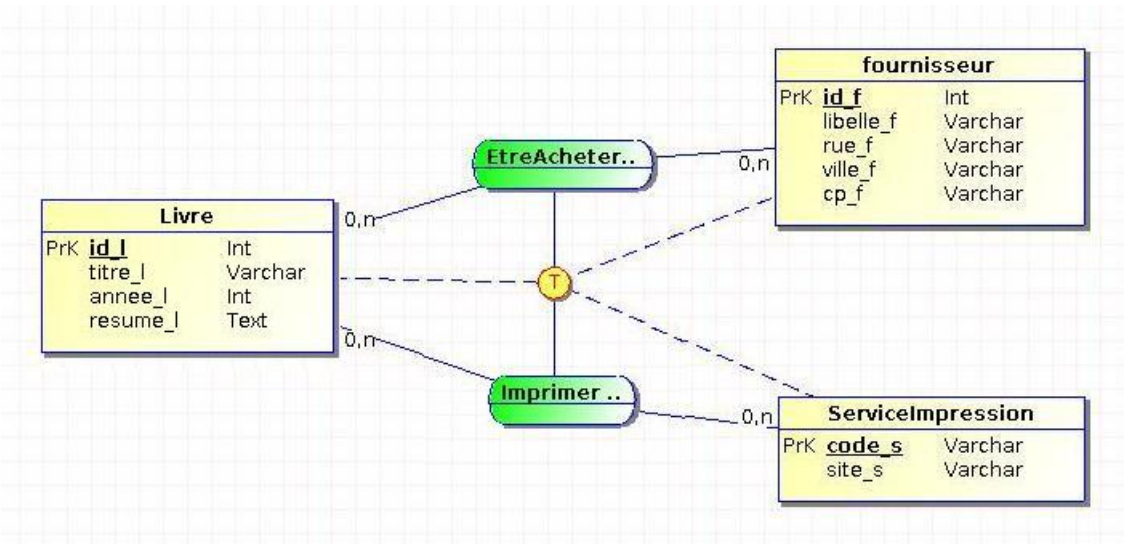


Dans cet exemple, un livre ne peut pas à la fois être acheté chez un fournisseur et être imprimé par un service d'impression interne.

IV-C-3 - La contrainte de totalité

Chacune des occurrences d'une entité doit être présente dans au moins une de ses associations qui font l'objet d'une contrainte de totalité.

Reprenons notre exemple précédent et adaptons-le à la contrainte de totalité :

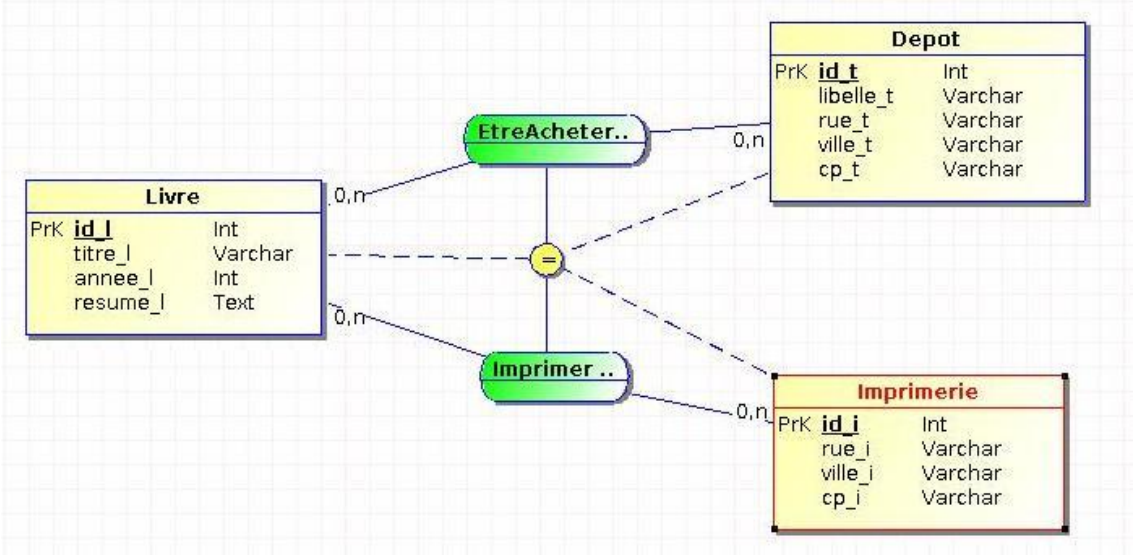


Dans cet exemple, un livre est toujours imprimé dans un service interne ou acheté par un fournisseur.

IV-C-4 - La contrainte d'égalité

Une occurrence présente dans une des associations concernées par la contrainte d'égalité l'est également dans toutes les autres associations concernées par cette contrainte.

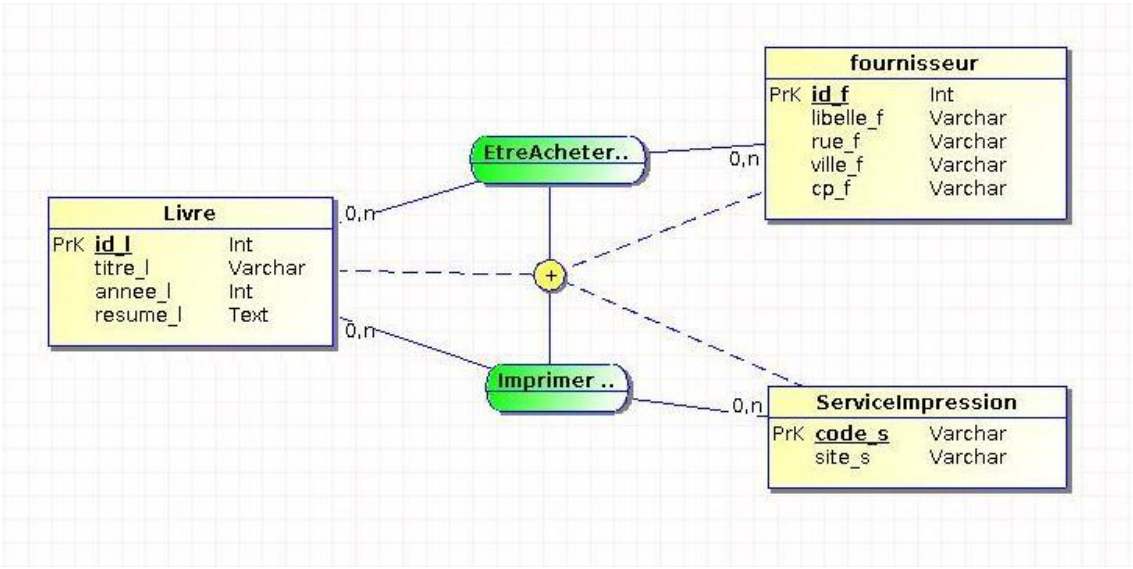
Prenons l'exemple d'un magasin qui vend des livres et qui souhaite archiver les dépôts et les imprimeries du livre :



Un livre acheté dans un dépôt sera donc également imprimé dans une imprimerie et vice et versa.

IV-C-5 - Combinaison entre contraintes

Comme dans le cadre d'un héritage, il est possible de combiner les contraintes (TI, T=, XT ou + pour la partition, etc). Voici un exemple de contrainte de partition :

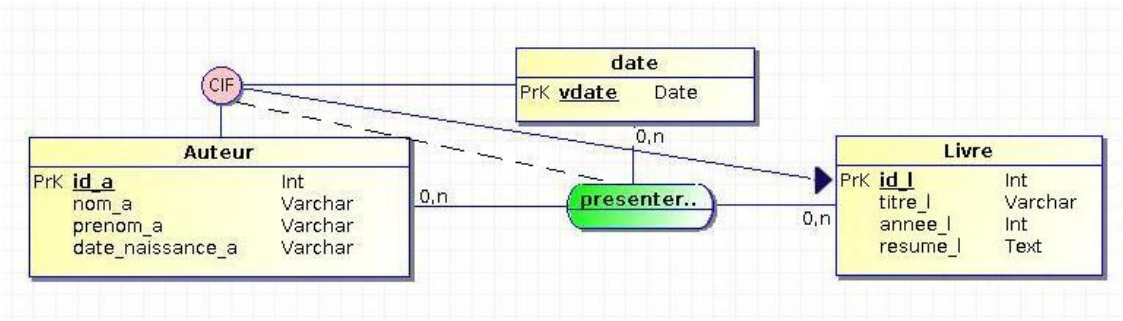


Pour cet exemple, le livre sera soit imprimé dans un service interne, soit acheté.

IV-D - Les CIF (contraintes d'intégrités fonctionnelles) et agrégations

Il s'agit de dépendances fonctionnelles qui sont directement représentées sur le MCD afin de réduire les identifiants d'associations jugés «trop larges». Ces DF sont des règles de gestion à faire apparaître sur votre schéma.

Reprenons l'exemple de l'auteur qui fait la promotion de son ouvrage à une date donnée :



Cela se traduit par la dépendance fonctionnelle suivante :

id_a, vdate → id_l

Ainsi, l'association «presenter» serait implantée comme ceci au niveau relationnel :

Presenter (*id_a#*, vdate, *id_l#*)
Légende :
x : relation
x : clef primaire
x# : clef étrangère

L'identifiant du livre ne fait donc plus partie de la clef primaire afin de garder une dépendance fonctionnelle directe et élémentaire.

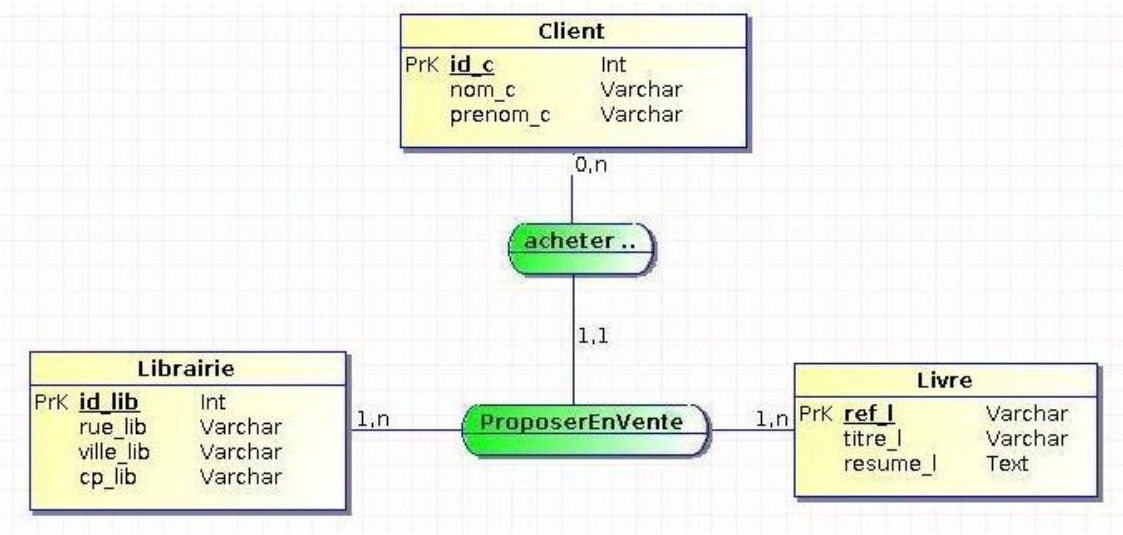
Cette notion de dépendance fonctionnelle peut être aussi représentée sous la forme d'agrégation (encore appelée «association d'association»).

Imaginons par exemple qu'une règle de gestion nous impose le fait qu'un livre, pour une librairie donnée, ne puisse être acheté que par un et un seul client.

La règle de gestion serait ici implantée par la dépendance fonctionnelle suivante :

id_lib, ref_l → id_c

Cette DF pourrait très bien faire l'objet d'une CIF au niveau conceptuel, où bien être représentée sous cette forme :



Voici comment cela se traduirait au niveau relationnel :

Librairie (id_lib, rue_lib, ville_l, cp_l)
Livre (ref_l, titre_l, resume_l)
Client (id_c, nom_c, prenom_c)
ProposerEnVente (*id_lib#*, *ref_l#*, *id_c#*)
Légende :
x : relation
x : clef primaire
x# : clef étrangère

On retrouve bien la même implantation au niveau relationnel que l'on aurait eu pour une CIF.

V - Conclusion

Avec les différentes notions abordées dans ce cours et quelques connaissance en SQL, il vous est maintenant possible de concevoir et réaliser des bases de données relationnelles. Il est important de maîtriser les différentes notions de ce modèle qui est aujourd'hui le plus enseigné au niveau des formations, mais aussi le plus utilisé en entreprise.

Cependant, ce modèle a souvent montré ses limites au niveau de certains systèmes d'informations. Par exemple, vous pourriez être amenés à travailler sur des systèmes d'informations où les SGBDR ne servent qu'à persister des données au format XML. Le traitement de l'information, qui doit parfois se faire en temps réel, se fait donc au niveau de l'application afin de ne pas surcharger de requêtes les serveurs de données.

VI - Remerciements

Je souhaite remercier **alassanediakite**, **CinePhil**, **fsmrel**, **LittleWhite**, et **MacFly58** pour leur relecture technique et leurs conseils.

Je tiens aussi à remercier **Erielle** pour son effort de relecture orthographique.