Petit cours de Modélisation

Introduction à Merise avec exercices et corrigés

Enseigné dans le M2 PISE de l'Université Paris 7 par Christophe DARMANGEAT.

Toutes les erreurs, insuffisances et plaisanteries affligeantes contenues dans ce site relèvent de la seule responsabilité de la bassesse de leur auteur (NB : attention, jeu de mots).

Si ce cours vous a instruit, ou amusé, ou mieux, les deux à la fois, n'hésitez surtout pas à me remercier en m'invitant au restaurant ou sur une barrière de corail. Ou mieux, les deux à la fois.

3 – Merise et le diagramme « entités-relations »

Merise est le nom d'une méthode, ou d'un ensemble de méthodes, développée en France dans les années 1970, et qui a été très largement employée depuis lors. Depuis une quinzaine d'années, Merise laisse peu à peu place à UML une autre norme nous n'aborderons pas ici (mais vous aurez tout de même bien du mal à y échapper).

Les puristes qui tomberont sur ce cours seront sans doute horrifiés devant la manière dont il taille dans le vif du sujet : loin de faire ici une véritable présentation de Merise, il s'agit simplement, si l'on veut, d'une introduction au début du commencement d''une esquisse d'une approche de Merise.

L'auteur de ces lignes revendique pleinement ses choix. D'abord, parce que ce cours s'adresse à de complets débutants en informatique, et qu'il faut bien commencer par quelque part. Ensuite, parce que le volume horaire dudit cours ne permet pas d'aborder autre chose que les grands fondamentaux – et comme dans tout apprentissage raisonné, mieux vaut moins, mais mieux. Enfin, parce que le propos de ce cours est bien moins de faire ingurgiter une série de conventions plus ou moins arbitraires que de permettre à ceux qui le suivront de s'approprier les raisonnements qui se cachent derrière.

3.1 – Le cadre général

Merise constitue donc un ensemble très riche de méthodes et de représentations, dont nous ne verrons ici qu'une petite partie - mais la plus cruciale.

Toute base de données va donner lieu à une **double représentation** :

1. le plan le plus abstrait (mais qui contient déjà toutes les informations indispensables pour la construction de la base de données) . Ce plan porte le nom de fort poétique de Modèle Conceptuel de Données (MCD).
2. un plan plus proche de ce que sera la base effective, telle qu'elle sera réalisée sur machine : le Modèle Logique des Données (MLD).

Le point crucial à enregistrer dès maintenant, c'est que **le MLD se déduit strictement du MCD d'après des règles formelles**. Autrement dit, une fois le MCD réalisé, il n'y a plus besoin de réfléchir une seule seconde pour produire le MLD : tout se fait par automatismes. La meilleure preuve, c'est qu'il existe des logiciels qui se proposent de réaliser le MLD d'un clic de souris, d'après le MCD. En revanche, il n'existe rien de tel pour concevoir le MCD : le seul ingrédient qui entre dans sa composition est l'huile de neurones. N'oubliez pas de faire quelques provisions...

3.2 – Le MCD, les entités et les relations

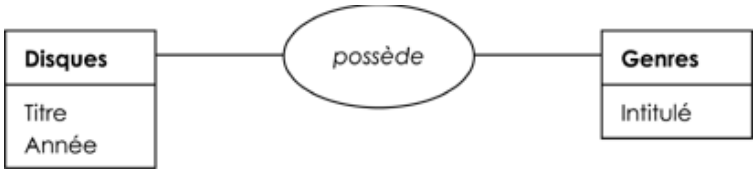
Les informations à traiter doivent être regroupées en ensembles cohérents, comme dans les tableaux que nous avons constitués il y a un instant. Dans les conventions de Merise, ces ensembles s'appellent des entités, et sont symbolisés par des rectangles. Chaque entité porte un nom, qui l'identifie de manière unique. Ce nom sera obligatoirement un **substantif au pluriel** : pour notre discothèque, on propose très logiquement « Disques » et « Genres ».

Les entités comprennent toujours un certain nombre d'éléments appelés propriétés (on parlera aussi d'attributs). Il s'agit des différentes rubriques qui devront être renseignées pour chaque individu.

Chaque entité, lorsqu'on passera au MLD (puis à la réalisation concrète de la base) donnera lieu à un tableau (on parle plus volontiers de tables). L'entité Disques du MCD produira donc une table Disques dans le MLD, et l'entité Genres, une table Genre. Les différentes propriétés de l'entité, qui sont donc écrites les unes sous les autres, deviendront **les titres des colonnes** de ces tables. Et dans ces colonnes, on fera figurer les différentes valeurs que prennent ces propriétés pour chacun des éléments de nos tables. Si vous trouvez cette explication un peu compliquée, pensez tout simplement à l'entité / table Disques : le titre, l'année et l'artiste sont disposés les uns sous les autres lorsqu'on parle de l'entité, et les uns à côté des autres (ce sont des en-têtes de colonne) lorsqu'on la représente comme une table.

Il ne reste plus à signifier que pour que chaque CD possède un genre (et pas n'importe lequel), mes deux entités doivent se trouver en relation l'une avec l'autre. Cette relation (on peut aussi parler d'association) sera symbolisée par un ovale, et sera nommée (par un **verbe**).

Voilà donc ce que cela donne :



Cette représentation ne se lit pas n'importe comment. Pour être certain de ne pas commettre de contresens, lorsqu'on traduit le schéma ci-dessus, il vaut mieux éviter de dire « *les disques possèdent des genres* », ou pire encore « *la table disque possède certains genres* » La bonne traduction, celle qui vous évitera au maximum de commettre des erreurs, consiste à dire que « **Chaque élément de la table Disques possède un (ou plusieurs ?) genres** ». En prenant l'affaire par l'autre bout, on peut tout aussi bien dire (même si c'est un peu laid à l'oreille) : « Chaque genre est possédé par un (ou plusieurs ?) disques » (on verra un peu plus loin comment en avoir le cœur net sur ces points d'interrogation).

Résumons-nous :

- les données doivent être systématiquement regroupées de manière à éviter les redondances, source de gâchis et, surtout, d'erreurs. Ce regroupement est la base de la modélisation.
- chacun des groupements (correspond, dans le MCD, à une entité, qui se traduira par une table dans la Base de Données.
- toute entité est symbolisée par un rectangle. Elle est nommée par un substantif au pluriel, désignant les éléments qu'elle contient
- les propriétés d'une entité correspondent aux colonnes de la table qui sera déduite de cette entité.

- les entités (c'est-à-dire : les individus présents dans les entités) peuvent être mises en rapport via des relations (ou associations)
- toute relation est symbolisée par un ovale. Elle est nommée par un verbe.

> **Remarque capitale :** l'étape consistant à créer autant d'entités que nécessaire est la première de toutes, et elle est absolument fondamentale. Le critère est simple : **aucune valeur de propriété ne doit se répéter** (sauf peut-être, à titre exceptionnel) **dans aucune table**. Cette règle, intangible quand il s'agit de valeurs de type texte, peut être assouplie lorsqu'il s'agit de valeurs numériques (dont les dates) : il serait en effet un peu ballot d'économiser la répétition d'un nombre... au prix de la création d'un code permettant d'accéder à ce nombre : ce serait payer de la main gauche ce qu'on économise de la main droite. Cependant, créer une entité composée uniquement de nombres peut se justifier, lorsqu'on veut restreindre les possibilités à un certain ensemble de valeurs (par exemple, les différentes motorisations disponibles pour une automobile).

3.3 – Éléments supplémentaires

3.3.1 – Deux nouveaux mots de vocabulaire

Dans une table, on évite de parler de « lignes » et de « colonnes ».

- Les lignes correspondent aux différents individus, ou aux différents objets individuels, répertoriés dans une table : dans la table Disques, chaque ligne correspond à un de mes CD. Ces différents éléments individuels qui correspondent aux lignes sont appelés enregistrements.
- Les colonnes, qui correspondent aux propriétés de l'entité dans le MCD, sont appelées des champs.

3.3.2 – Typage des propriétés

Vous ne serez guère étonnés d'appendre que les informations contenues dans les entités (donc, dans les tables) vont devoir au bout du compte être codées numériquement afin d'être stockées sur un support informatique, sous un nom qui est celui de la propriété.

Pour chaque enregistrement, celle-ci se comporte donc comme un nom de variable… ce qui est somme toute logique, car à de menus détails près, c'en est une. Tout ceci nous amène au fait que les propriétés, à l'instar des variables, relèvent de certains types.

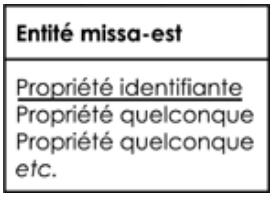
Dans le détail, les types disponibles pour les propriétés varient légèrement d'un système de gestion de bases de données à l'autre. En ce qui nous concerne, nous pouvons en rester à un niveau assez général, en considérant les types les plus courants :

- numérique : on y distingue systématiquement l'entier (« Integer ») du nombre à virgule (« Float »). Le type « AutoIncrement », souvent utilisé pour gérer les clés primaires [MODE NO PANIC ON] vous saurez ce que c'est très bientôt [MODE NO PANIC OFF], correspond à un entier dont la valeur est automatiquement attribuée à la création d'un nouvel enregistrement. Les bases de données proposent également toujours au moins un type Date/Heure.
- texte : on aura éventuellement différents types correspondant à différentes longueurs maximales du texte.
- booléen : inévitable !

Outre les informations précédemment citées, les documents de modélisation, MCD et MLD, devront donc faire apparaître, pour chaque entité, le type de chaque propriété.

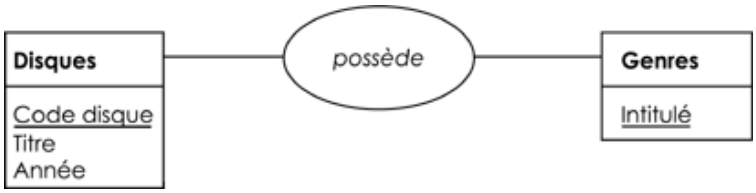
3.3 – Propriété identifiante (clé primaire)

Tout système de bases de données impose dans chaque entité, chaque individu (chaque enregistrement) puisse être identifié de manière unique, sans ambiguité, par la machine. Le procédé le plus courant consiste à dédier à cela une propriété spéciale, appelée propriété identifiante ou encore clé primaire. On peut constituer une clé primaire à partir d'une combinaison de champs, mais nous verrons que c'est une solution qui n'est employée que dans certains cas particuliers ; restons-en donc pour le moment à l'idée que la clé primaire est un champ spécial. La clé primaire est alors généralement placée en tête de la liste des propriétés, en la soulignant pour indiquer son statut particulier :



Il est en fait assez rare de trouver spontanément une propriété capable de jouer ce rôle. Même les propriétés qui semblent faire de bonnes candidates (par exemple, une plaque d'immatriculation ou un numéro de sécurité sociale) ne sont pas forcément aussi opportuns qu'elles en ont l'air, pour un certain nombre de raisons. Et il n'est pas rare qu'aucune des propriétés présentes ne puisse nous prémunir contre les doublons ; c'est le cas avec l'entité *Disques* de notre exemple : plusieurs Cd peuvent très bien avoir le même titre, et je ne parle pas de l'auteur ni de l'année. On ne peut pas davantage exclure la possibilité que deux auteurs homonymes aient sorti la même année un disque portant le même titre (ce qui nous empêche donc d'avoir confiance dans une clé primaire constituée de la combinaison des trois propriétés).

Voilà pourquoi le plus souvent, on sera amené à créer une **propriété supplémentaire** destinée uniquement à jouer le rôle d'indentifiant / clé primaire. Il s'agira presque toujours d'un code, unique pour chaque occurrence de l'entité (et voilà pourquoi un nombre de type « autoincrement » est si pratique). Ce code sera rarement visible par l'utilisateur, qui ignorera sans doute son existence : il n'en sera pas moins indispensable pour le système informatique. Ainsi, notre modèle de discothèque deviendra-t-il :



3.4 – Les cardinalités

Contrairement à ce que certains pourraient penser, ce terme n'indique ni le fait de devenir cardinal (Dieu m'en garde !) ni le fait de se situer à l'un des quatre points cardinaux (en l'occurrence, complètement à l'Ouest). Non, la cardinalité, c'est un mot savant de mathématicien pour dire tout bêtement que l'affaire a un rapport avec des nombres et des quantités. Pour dire la même chose, on aurait pu donc tout simplement parler de « quantité »

ou de « numération ». Sauf que dans un dîner de famille, placez subrepticement « cardinalité » entre la poire et le fromage, et vous pourrez vérifier par vous-mêmes que l'effet obtenu n'est pas du tout le même.

3.4.1 – Encore et à nouveau la discothèque

Dans le MCD que l'on vient d'élaborer, il manque une information essentielle pour la suite : à combien d'éléments de l'autre entité chaque élément peut-il être associé ? Lorsqu'on bâtit une relation entre deux entités, on doit nécessairement préciser ce point, car de lui dépendent de très importantes conséquences.

Dans l'exemple que nous avons pris, celui de la discothèque, il paraît évident qu'un même genre musical peut être représenté par plusieurs disques. En sens inverse, en revanche, on peut être embêté pour décider à combien de genres peut correspondre chaque disque. On peut en effet imaginer soit que chaque disque ne puisse être rattaché qu'à un genre et un seul afin de faciliter le classement, soit qu'à chaque disque on puisse attribuer plusieurs genres à la fois, ce qui introduit davantage de complexité, mais aussi davantage de souplesse.

La décision, dans cette alternative, n'appartient pas à l'informaticien : il s'agit d'un choix d'ordre fonctionnel, qui doit être subordonné aux besoins de l'organisation pour laquelle est conçue de la base de données. Il n'existe donc aucune règle qui permette de trancher a priori entre les deux possibilités... excepté que la technique doit être au service des besoins de ceux qui s'en serviront, et non l'inverse. En revanche, ce qui nous intéresse ici, ce sont les conséquences de ce choix sur notre base de données.

3.4.2 – Cardinalités minimum et maximum

Un modèle (conceptuel), lorsqu'il met en relation deux entités A et B, doit toujours stipuler à combien d'éléments de l'entité B chaque élément de A peut correspondre, et inversement – c'est ce qu'on appelle la définition des cardinalités. De là, il faut distinguer le nombre minimum et le nombre maximum de ces correspondances : pour chaque élément d'une entité, on doit donc stipuler à combien d'éléments de l'autre entité celui-ci est susceptible de correspondre, au minimum et au maximum. Ainsi, toute relation entre deux entités impose de préciser quatre nombres (quatre cardinalités) : cardinalité minimum de A vers B, maximum de A vers B, minimum de B vers A, maximum de B vers A.

En reprenant l'exemple de la dicothèque, cela revient à se poser les questions suivantes :

- À combien de genres au minimum correspond chaque CD ? (autrement dit : un CD peut-il ne pas avoir de genre, ou en a-t-il forcément au moins un ?)
- À combien de genres au maximum correspond chaque CD ? (autrement dit : un CD peut-il avoir plusieurs genres, ou est-il limité à un seul ?)
- À combien de CD au minimum correspond chaque genre ? (autrement dit : ma table des genres comprend-elle uniquement des genres qui correspondent à mes CD, ou peut-il y avoir des genres « orphelins » ?)
- À combien de CD au maximum correspond chaque genre ? (autrement dit : puis-je avoir plusieurs CD du même genre, ou est-ce interdit ?)

3.4.3 – Les valeurs possibles

Les cardinalités obéissent à un formalisme assez étroit :

- La cardinalité minimum ne peut prendre que les valeurs 0 ou 1. Autrement dit, soit on considère qu'un élément de la table A *peut* être en relation avec un (ou plusieurs) éléments de la table B (mais que ce n'est pas obligatoire), soit on considère que tout élément de la table A doit impérativement être en relation avec au moins un élément de la table B. Dans notre exemple, choisir 1 comme cardinalité minimum signifie qu'un

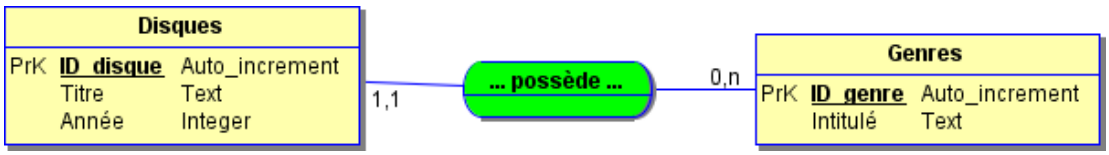
disque doit etre classé dans au moins un genre. Choisir 0 signifie qu'on estime que certains disques n'ont pas forcément de genre (le même problème se pose dans l'autre sens de la relation, pour savoir s'il peut exister ou non des genres sans disques).

- La cardinalité maximum ne peut valoir que 1 ou N, autrement dit un ou plusieurs. C'est la discussion de tout à l'heure à propos des genres : autorise-t-on ou non chaque CD à être classé dans plusieurs genres à la fois ? En sens inverse, il ne fait aucun doute qu'à chaque genre, doivent pouvoir correspondre plusieurs CD.

Il n'existe donc que quatre cas de figure possibles pour les cardinalités : (0, 1), (0, N), (1, 1) et (1, N).

3.4.4 – Représentation

Dernier point, les cardinalités minimum et maximum sont représentées sous la forme d'un couple de nombres placé entre l'entité et la relation. Par exemple, dans le cas de notre discothèque, si on limite à un seul le nombre de genres autorisés par disque et qu'on conserve des genres sans disques correspondants, on a :



3.4.5 – Raccourcis de langage

Pour décrire les cardinalités, on va souvent user d'un raccourci de langage. Comme les cardinalités les plus décisives sur l'architecture de la base sont souvent les cardinalités maxima, on aura tendance à ne parler que d'elles. Ainsi, dans le cas d'une relation où l'une des deux cardinalités maximales vaut 1 et l'autre N, on dira volontiers qu'on a affaire à une relation « un à plusieurs ». Lorsque les deux cardinalités maximales valent N, on parlera de relation « plusieurs à plusieurs ».

> Une relation de type « un à un » (où, donc, les deux cardinalités maximum sont égales à 1) est un cas limite. Cela signifie que nous avons créé deux entités qui en réalité n'en forment qu'une seule, puisque chaque élément de l'une corespond à un élément de l'autre, et à un seul. Ce n'est pas à proprement parler une faute, mais face à une telle situation, on a toujours intérêt à se demander ce qui justifie d'avoir créé deux entités plutôt qu'une seule.

3.5 – Du MCD au MLD : première approche

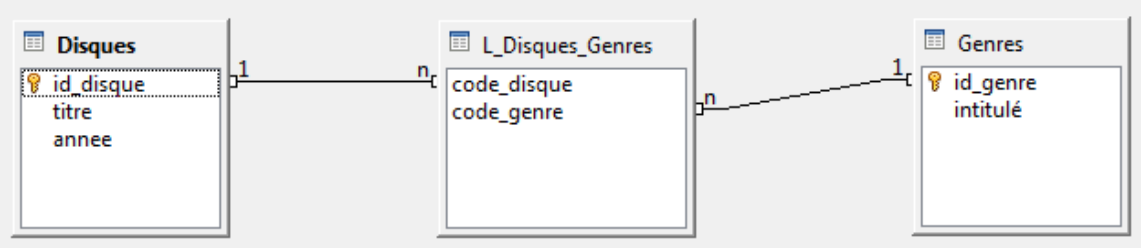
Je rappelle que par rapport au MCD, le MLD est un plan moins abstrait et plus proche de la réalité, c'est-à-dire de la base de données telle qu'elle existera sur les machines. Je rappelle aussi que toute la difficulté de la modélisation réside dans l'élaboration du MCD. Une fois que celui est conçu, le MLD s'en déduit par l'application de quelques règles (on pourrait dire : d'un algorithme). Donc, le passage du MCD au MLD n'est qu'une question de rigueur, et plus du tout d'intelligence ou d'imagination (« surtout pas ! », pourrait-on même dire).

Une première règle, d'une simplicité biblique, est que **toute entité du MCD devient une table du MLD. L'identifiant de chaque entité devient la clé primaire de chaque table.**

Ensuite, selon les cardinalités maximales qui caractérisent la relation, les choses vont se passer très différemment.

3.5.1 Si les deux cardinalités maximales sont N

...autrement dit, si la relation est de type « plusieurs à plusieurs ». Dans le MLD, la relation devient alors une **nouvelle table**, elle-même en relation avec les deux tables produites par les deux entités. Une telle table est dite table de correspondance, ou encore table de liaison, table de jonction, table d'association, etc. Elle ne contient pas à proprement parler des données : son rôle est d'organiser les rapports entre les éléments des tables qui, elles, les contiennent. Une table de jonction contiendra uniquement des propriétés correspondant aux clés primaires des deux entités, qu'elle associera deux à deux :



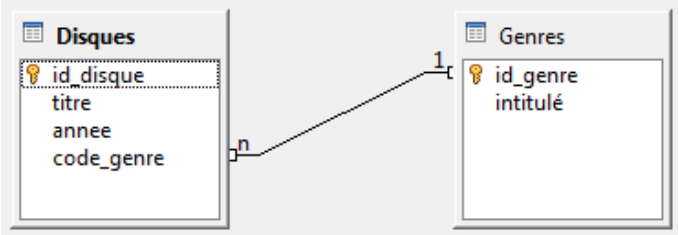
C'est très logique : la table de jonction va permettre d'associer tout élément de Disques à tout élément de Genres autant de fois que souhaité. Un même disque pourra ainsi être mis en rapport avec plusieurs genres, et un même genre avec plusieurs disques.

3.5.2 Si l'une des deux cardinalités maximales vaut 1

Ce cas se divise à son tour en deux, selon la valeur de la cardinalité minimale concernée.

1. **On a une cardinalité (1, 1)**

   Concrètement, cela veut dire, dans notre exemple, que chaque CD possède un genre et un seul. Un CD ne peut pas ne pas avoir de genre ; il ne peut pas non plus en avoir plusieurs. Dans ce cas, dans le MLD, cette relation devient une relation **directe** entre les deux tables. L'identifiant de la table côté « plusieurs » devient une nouvelle propriété de la table du côté « un », ainsi que l'illustre cet exemple :



   Quand on y réfléchit, c'est parfaitement logique : pour chaque disque, il faudra renseigner un code (et un seul) qui correspondra à l'un des genres présents dans la table Genres. La table Genre, elle, ne contient aucun code renvoyant vers les disques - sinon, cela voudrait dire que chaque Genre renvoie vers un disque et un seul ! En fait, on vient de retrouver par un long détour l'exemple à partrir duquel on avait introduit le raisonnement (« Tout ça pour ça », direz-vous...)

   Et, au passage, pour cette nouvelle propriété dans la table Disques qui contiendra une valeur prise par la clé primaire de la table Genre, on dit qu'il s'agit d'une clé étrangère.

2. **On a une cardinalité (0, 1)**

   > **Remarque essentielle :** le raisonnement qui suit concerne en réalité **toute** situation dans laquelle la cardinalité minimale est égale à zéro. Cela correspond à la situation où chaque CD peut avoir un genre (au maximum) mais où il n'est pas obligé d'en avoir un. Là, les informaticiens se divisent en deux catégories. Il y a les coulants (les autres préfèreront les appeler les laxistes), qui diront : « Faisons au plus simple. Il suffit de créer une clé étrangère ; lorsque le disque n'aura pas de

genre attribué, la valeur de la clé étrangère sera vide. Après tout, il n'y a pas de mal à cela. » Au passage, une valeur vide, dans une base de données, s'appelle en jargon une valeur Null. À cela, l'autre catégorie d'informaticiens, beaucoup plus nombreuse, que sont les rigoureux (et que les premiers appelleront les psycho-rigides) rétorquent : « Certes, on peut avoir des valeurs Null dans une table. Mais ce n'est jamais une bonne chose. On ne sait pas, par exemple, si c'est un défaut de saisie ou une valeur volontaire. Et puis, quand on fera des recherches ou des traitements automatisés, cela risque de nous jouer de bien vilains tours. Il est donc nettement préférable de jouer la sécurité : il suffit de créer une table de jointure, exactement comme dans le cas d'une relation « plusieurs à plusieurs ». Dans ce cas, plus de Null ; en revanche, il faudra mettre en place un contrôle pour être certain que chaque disque n'apparaît pas plus d'une fois dans la table d'association...

On remarquera qu'en ce qui concerne les normes de représentation des cardinalités, c'est... la jungle. Les images ci-dessus, réalisées avec LibreOffice Base, placent en effet dans le MLD le « 1 » du côté de la clé primaire et le « plusieurs » du côté du code qui pointe vers celle-ci, soit exactement l'inverse de ce qu'impose Merise dans le MCD. Bon, d'un autre côté, comme Merise fait pour sa part exactement le contraire d'UML, on peut dire en quelque sorte que quelque part (mais où ?), on retombe sur nos pattes. Menfin, la seule vraie conclusion à tirer est que le positionnement des cardinalités relève de la pure convention, pour ne pas dire de l'arbitraire le plus total.

Eh bien avec tout ça, il est grand temps de passer à la pratique :

exercice 1
exercice 2