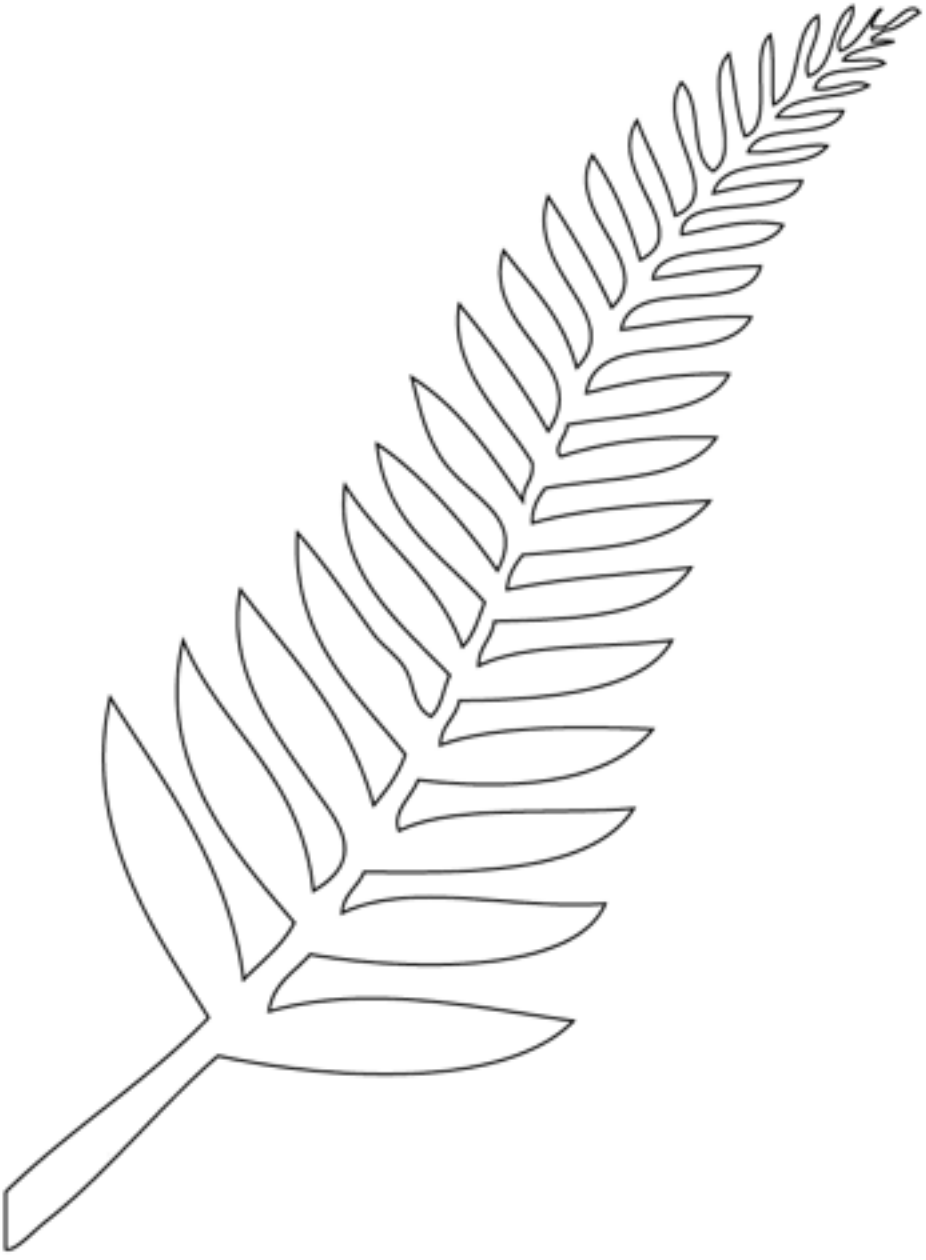
BASE DE DONNÉES
modélisation

Définitions....2
Modélisation des processus: notions....3
Premiers problèmes… et premières solutions....8
Diagramme “entités-relations”....12
La normalisation du modèle....20
Techniques avancées....22
Situations embarrassantes....26



DÉFINITIONS

Une base de données (BD) est un ensemble de données mémorisées sur des supports accessibles par un ordinateur pour satisfaire simultanément plusieurs utilisateurs de façon sélective et en temps très court. Elles constituent le cœur du système d'information.

Il existe 4 types de bases de données :

1. **Hiérarchiques**: les plus anciennes fondées sur une modélisation arborescente des données.
2. **Relationnelles**: organisation des données sous forme de tables et exploitation à l'aide d'un langage déclaratif (ex: Oracle, mySQL, Access).
3. **Déductives**: organisation de données sous forme de table et exploitation à l'aide d'un langage logique.
4. **Objets**: organisation des données sous forme d'instances de classes hiérarchisées qui possèdent leur propres méthodes d'exploitation.

Dans les lignes qui suivent nous aborderons la conception et la mise en place de bases de données relationnelles. La conception de ces bases est la tâche la plus ardue du processus de développement du système d'information. Les méthodes de conception préconisent une démarche en étapes et font appel à des modèles pour représenter les objets qui composent les systèmes d'information, les relations existantes entre ces objets ainsi que les règles sous-jacentes.

Modéliser, kesako ?

Commençons par un scoop: un système d'information a pour objectif... de traiter des informations. Autrement dit, son action va consister à accéder à des informations (qui auront donc été plus ou moins convenablement rangées), puis à les afficher et/ou à les modifier, avant de les re-ranger pour une utilisation ultérieure. Concevoir un système d'information, c'est donc concevoir tout à la fois:

- Les traitements proprement dits: quelles données doit-on aller chercher à quel moment, et comment doivent-elles éventuellement être modifiées ?
- La manière dont les données vont être organisées: quelles informations doivent-elles être regroupées ou, au contraire, séparées ? Comment une information permettra-t-elle d'en retrouver une autre ? Etc.

Selon les tâches à effectuer, un système d'information mettra davantage l'accent sur l'un ou l'autre aspect. Un système d'information à visées scientifiques, par exemple, fera sans doute beaucoup de calculs compliqués à partir d'un nombre de données pas forcément très grand. C'est donc la partie traitements qui sera la plus importante. Inversement, la consultation d'un catalogue de pièces détachées ou d'un annuaire nécessitera un minimum de traitements, mais la qualité du rangement des données y sera cruciale.

Il n'existe pas une frontière étanche entre les deux aspects: traiter les données peut inclure des opérations d'extraction et de rangement. Inversement, on ne peut se préoccuper d'organiser ses informations sans se poser un minimum de questions sur la manière d'y accéder, de les supprimer ou de les modifier. Même si elle donc pas absolue, la distinction est cependant pertinente, et permet de définir deux grands domaines de l'informatique:

- **Le monde des traitements est celui de la programmation**
  L'approche de la programmation à un niveau abstrait, consistant à en formuler la logique s'appelle l'algorithmique (un mot terrifiant qui recouvre une réalité bien plus terrifiante encore, comme vous pouvez vous en rendre compte en étudiant attentivement le cours que voici que voilà).
- **Le monde de l'organisation des données est celui des bases de données**
  (même si toutes les données d'un système d'information ne sont pas forcément regroupées dans une “base de données”, les bases de données rassemblent les informations dès lors que celles-ci se présentent en masse). L'approche abstraite de l'organisation des données dans une base s'appelle la modélisation.

La modélisation est donc à l'organisation des données ce que l'algorithmique est à leur traitement: l'approche la plus abstraite, le plan à partir duquel sera édifié le bâtiment réel.

MODÉLISATION DES PROCESSUS: NOTIONS

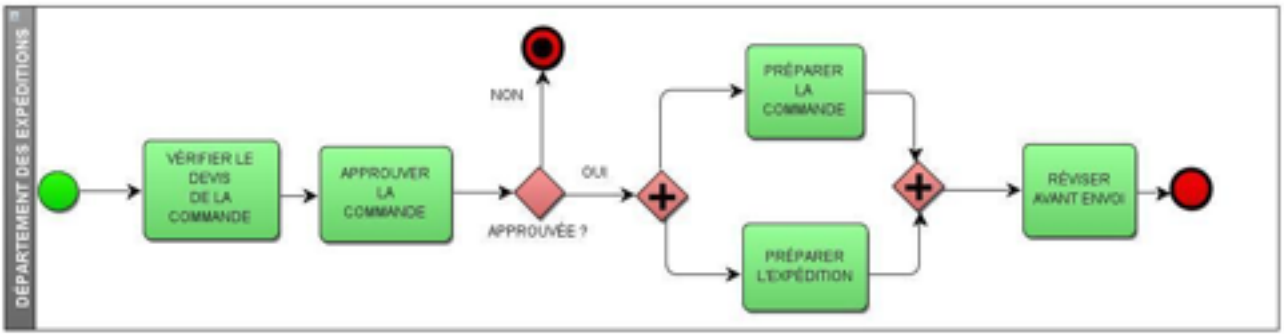
Modéliser permet, on vient de le voir, de faciliter la communication entre les différents acteurs engagés dans le développement et la maintenance de solutions, en particulier liées au processus métier des entreprises. En fait, par une représentation standardisée du déroulement des processus facilite, grâce au schéma la compréhension des besoins.

Exemple de modélisation 1: expédition des commandes

Nous voulons représenter le travail effectué par le département des expéditions lors du traitement d'une commande. Les opérations se déroulent comme suit :

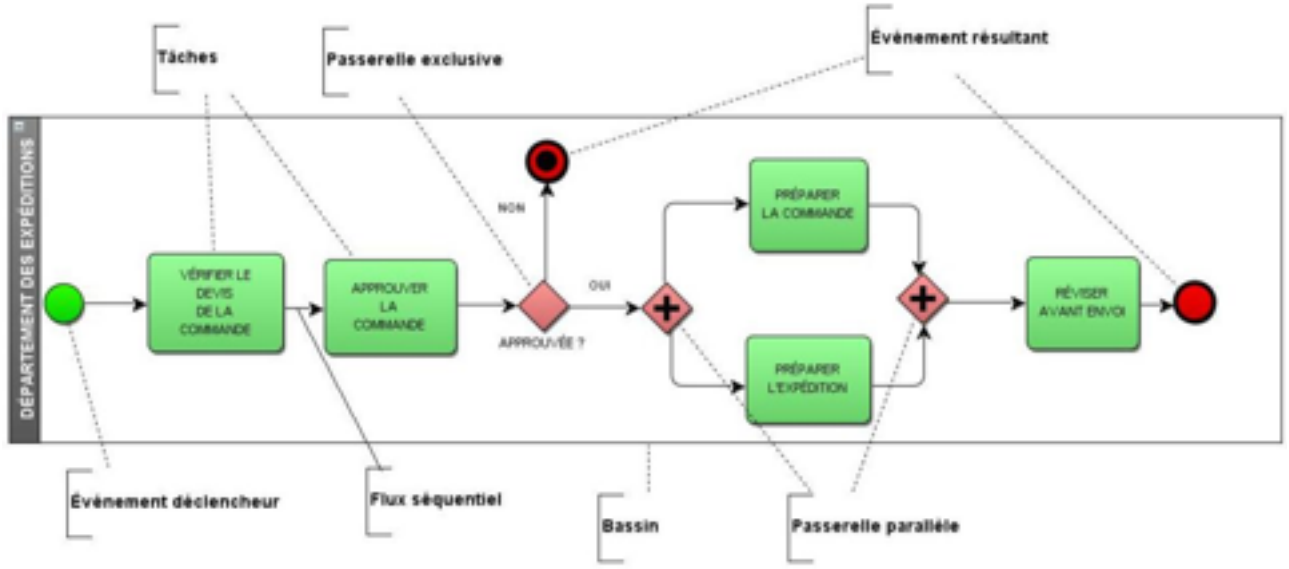
- Suite à la réception du devis de la commande, le responsable du département doit vérifier si les articles sont disponibles dans l'entrepôt.
- Si les articles sont disponibles, il approuve la commande. Dans le cas où il y a des articles en rupture de stock, il annule la commande et le processus se termine. (Il recevra éventuellement une nouvelle liste d'articles).
- Suite à l'approbation d'une commande, un commis prépare les articles de la commandes.Durant la même période, un autre commis prépare les modalités d'envoi avec la compagnie d'expédition.
- Une dernière révision est effectuée avant l'envoi afin de s'assurer que tout concorde.

Nous utiliserons ici une notation type BPMN pour Business Process Model and Notation.



Symboles de modélisation

Jetons un coup d'œil sur les symboles utilisés dans cette représentation du processus d'expédition des commandes.



Nous retrouvons :

- **Événement déclencheur** qui illustre le début ou le point de départ d'un processus
- **Tâches**, représenté par un rectangle, la tâche illustre une unité de travail
- **Passerelle** exclusive, elle s'apparente à une question que l'on pose à un certain moment au cours du processus
- **Flux** séquentiel, représenté par une flèche, il est utilisé afin d'illustrer l'ordre d'exécution des activités ou des tâches tout en reliant les symboles
- **Passerelle parallèle**, utilisée pour combiner des séquences qui peuvent se dérouler indépendamment l'une de l'autre
- **Événement résultant**, indique la fin du processus
- **Bassin**, représente le département qui s'occupe du processus ou les frontières des processus décrit

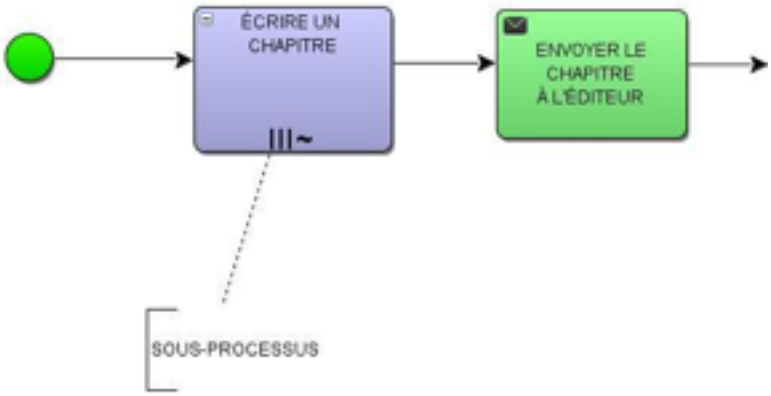
A noter également l'existence d'un symbole

- **"Couloir"**, qui représente la division d'un processus en sous-ensemble d'activités ou de tâches, par exemple dans le cas où les acteurs impliqués sont multiples
- "**Objet**" ou "Document", fournissant des informations sur les activités ou tâches qui doivent être réalisées

Description d'un sous-processus

Un sous-processus se compose de plusieurs tâches et est soumis aux mêmes règles qu'un processus normal. Si l'on veut modéliser plusieurs processus comportant plusieurs sous-processus, il convient d'illustrer les sous-processus sous forme de "zoom".

La figure ci-dessous illustre le sous-processus de rédaction d'un chapitre qui prend place dans un processus décrivant la rédaction d'un livre.



Exemple de modélisation 2: achat de fourniture

Une entreprise de location de matériel doit régulièrement se procurer de nouveaux produits.

Etape 1: identifier les acteurs (interne et externe)

Le processus implique quatre acteurs interne : le responsable de la division (patron), un responsable de contrat (coordonateur), un responsable des achats et le département de la comptabilité et un acteur externe : le fournisseur. Dans un premier temp, on se consacrera uniquement sur le processus privé c'est à dire, sur les opérations internes à l'entreprise.

Etape 2: définir les frontières du processus

Une fois les acteurs identifiés, on définit les frontières (ici représentées par les limites du bassin) du processus en dispensant des espaces, qui sont des couloirs, pour les participants.

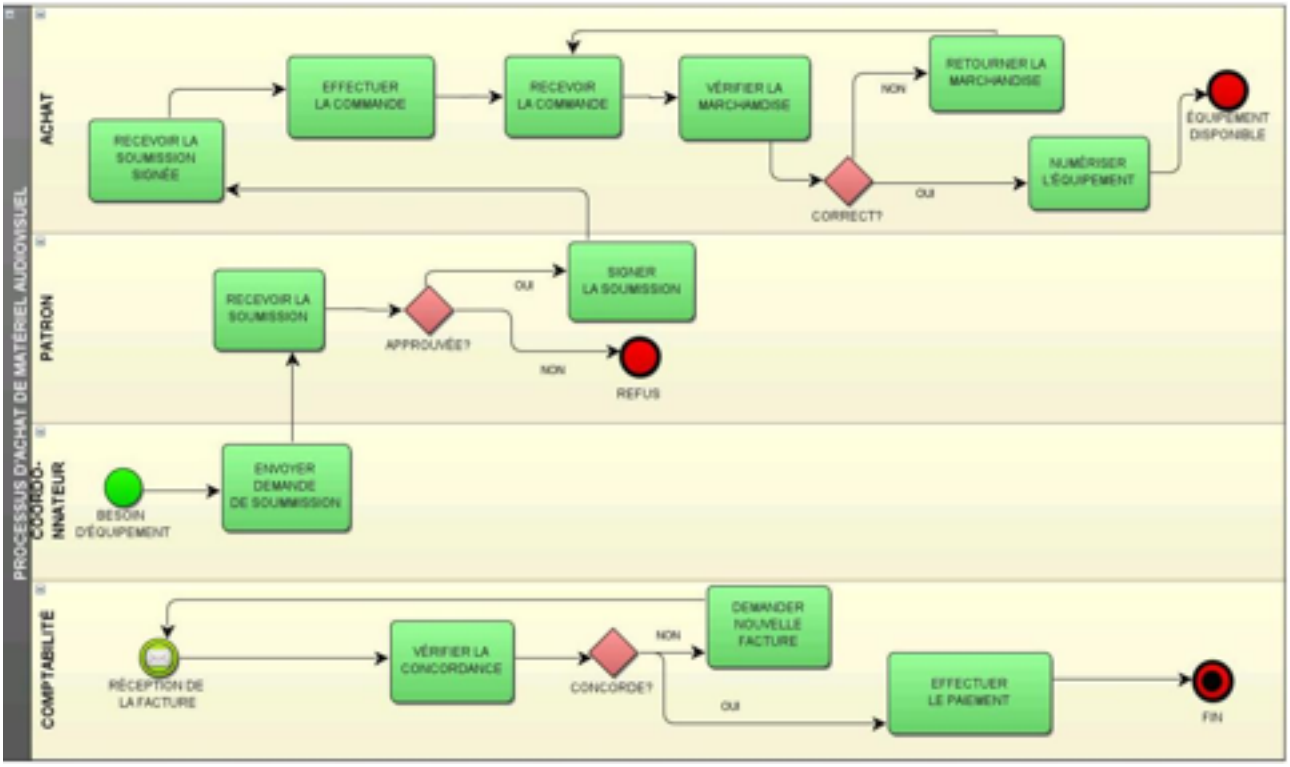
Etape 3: identifier les tâches

Au sein d'un processus privé, les opérations effectuées se doivent d'être détaillées. Nous définirons ainsi les tâches de chaque acteur.

Etape 4: identifier et représenter les événements à l'aide des symboles appropriés

On ajoutera les événements du processus. L'événement d'initialisation est le besoin émis par la personne qui gère les contrats. La réception de la facture est un événement intermédiaire. Le processus peut se terminer lorsque le patron refuse la soumission. La fin de séquence du responsable des achats est également considérée comme une finalité, car le matériel sera disponible et sa séquence d'activités terminée. Enfin, on reliera les activités et créera les points de branchements (passerelles) en fonction des différentes possibilités. On retrouve ici trois passerelles : lorsque le patron décide ou non de signer la soumission, lors de la réception et vérification si le produit est conforme et fonctionnel, et lorsque la

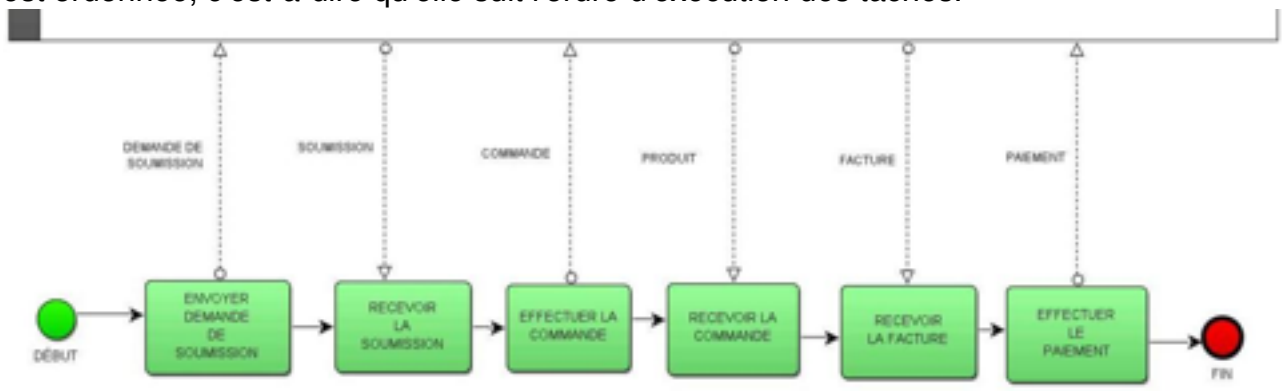
facture est envoyée, avec la vérification si elle concorde avec la soumission. L'ensemble nous donne un processus interne que nous nommerons privé.



Etape 5: modélisation du processus public

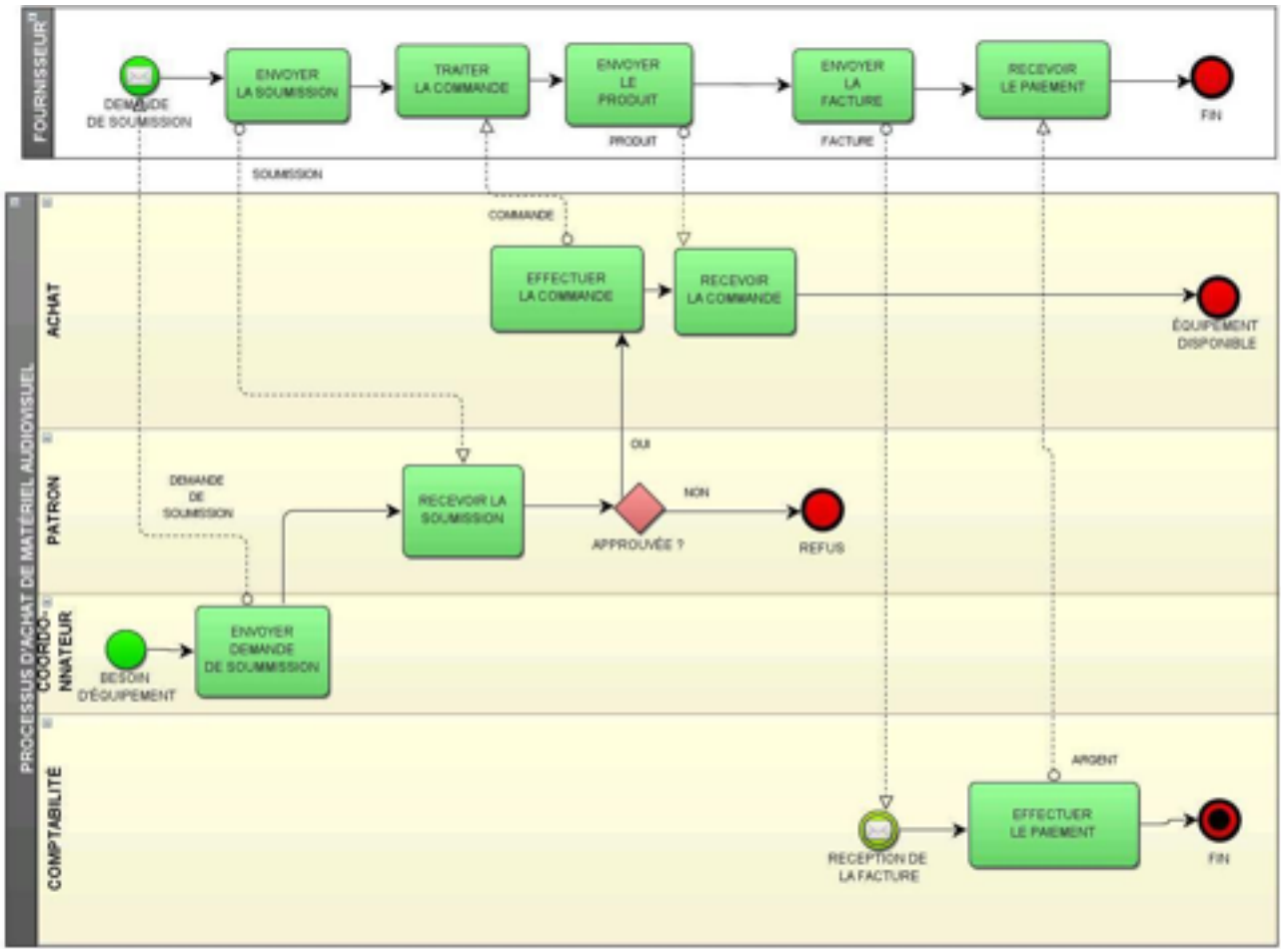
Suite à la modélisation du processus privé, nous retrouvons cinq tâches où l'entreprise interagit avec le fournisseur : envoyer demande de soumission, recevoir la soumission, effectuer la commande et effectuer le paiement. La réception de la facture est modélisée dans le processus privé comme un événement, que nous transposons en une tâche pour les biens de la représentation.

A noter que l'on retrouve, au sein du processus public, uniquement les tâches du processus privé qui nécessitent une communication, une réception ou un envoi d'information ou d'un produit physique. Il est important de s'assurer que la séquence du processus public est ordonnée, c'est-à-dire qu'elle suit l'ordre d'exécution des tâches.



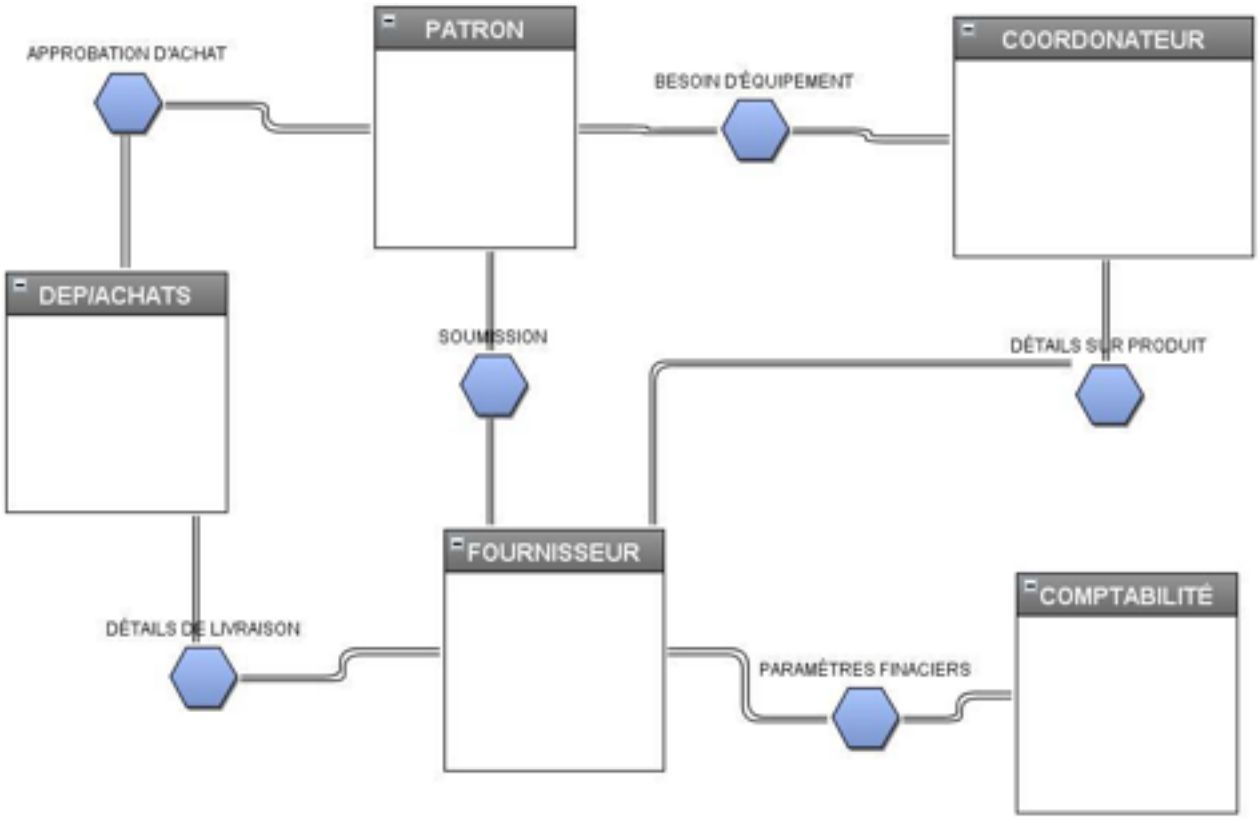
Etape 6: modélisation du processus de collaboration

Le processus de collaboration est issu des processus communiquant entre eux. Ici des séquences d'action avec un fournisseur. Le processus du fournisseur débute lorsque ce dernier reçoit une demande de soumission. Par la suite, nous savons que les tâches suivantes devront être effectuées : envoyer la soumission, traiter la commande, envoyer le produit, envoyer la facture et recevoir le paiement. Ces tâches comportent toutes des interactions avec le processus d'achat.



Etape 7: modélisation de conversation

Le diagramme de conversation nécessite que l'on modélise tous les intervenants comme des entités et que l'on joigne ces entités à l'aide du symbole de conversation que l'on nomme en désignant le sujet des interactions entre ces entités. Nous représentons ici les cinq participants du processus comme des entités soit : le patron, le coordonnateur, le département des achats, le fournisseur et la comptabilité.



Ce passage d'un ordinogramme au modèle de conversation constitue le début de la construction d'un modèle conceptuel de données (MCD) que nous étudierons un peu plus loin. Nous allons pour l'instant nous intéresser aux problèmes de la gestion et de l'organisation des données.

PREMIERS PROBLÈMES… ET PREMIÈRES SOLUTIONS

Organiser des données, où est le problème, direz-vous ? Y a-t-il vraiment besoin de faire des études pour saisir les informations liées à l'exemple précédent (achat de fournitures)? Il suffit d'être un peu soigneux, et il n'y a aucune raison que ça se passe mal!

Eh bien si, en fait. Les données, quand elles se présentent en (très) grandes quantités, posent des problèmes auxquels on ne s'attend pas. C'est bien pour cela qu'on a mis au point un certain nombre de règles et de méthodes qui, si elles ne peuvent dispenser personne de réfléchir, aident néanmoins à réfléchir én évitant les catastrophes les plus courantes.

Voyons les problèmes les plus évidents qui se posent lorsqu'on veut organiser des informations. Pour cela, imaginons que nous voulions informatiser les rayons du matériel audio présenté dans le chapitre précédent… – enfin, disons les dix premiers CD du rayonnage, parce que sinon, ça va être un peu long.

Faisons simple pour commencer: nous ne noterons, pour chaque CD, que le titre, l'année, le nom de l'artiste et le genre musical. Cela nous donne la liste suivante:

- Nursery Cryme, Genesis, 1972, rock progressif
- Foxtrot, Genesis, 1972, rock progressif
- Selling England by the Pound, Genesis, 1973, rock progressif
- Symphonie n°2, Sibelius, 1985, classique
- Symphonie n°7, Sibelius, 1987, classique
- Concerto pour violon, Mendelssohn, 1992, classique
- Crime passionnel, Guidoni, 1982, chanson française
- 5th Gear, Brad Paisley, 2007, country
- Thick as a Brick, Jethro Tull, 1973, rock progressif
- Purpendicular, Deep Purple, 1996, rock

Il va de soi que pour représenter de telles informations, une liste c'est bien, mais qu'un tableau, ce serait beaucoup mieux. Les conventions étant en accord avec ce que nous suggère l'intuition, on mettra en ligne, les uns en-dessous des autres, les différents disques (autrement dit, en quelque sorte, les "individus" de notre base). Et on portera en colonne les différents renseignements dont on dispose pour chacun de ces individus. Naturellement, il est préférable –- et, en réalité, obligatoire - de nommer ces colonnes: Ma discothèque prendrait ainsi la forme du tableau suivant:

TITRE	ARTISTE	ANNEE	GENRE
Nursery Cryme	Genesis	1972	rock progressif
Foxtrot	Genesis	1972	rock progresif
Selling England by the Pound	Genesis	1973	rock progressif
Symphonie n°2	Jean Sibelius	1985	classique
Symphonie n°7	Jean Sibelius	1987	classique
Concerto pour violon	Felix Mendelssohn	1992	Classique
Crime passionnel	Jean Guidoni	1982	chanson française
5th Gear	Brad Paisley	2007	country
Thick as a Brick	Jethro Tull	1973	rock progressif

Or, ce petit tableau, à lui seul, fait apparaître au moins deux problèmes majeurs.

1. Des informations identiques s'y répètent: en l'occurrence, celles concernant les artistes, et encore plus, le genre. En termes savants, on parle de redondance des informations. Imaginons que ma discothèque comporte plusieurs centaines de CD, il y a fort à parier que la mention "classique" ou "rock progressif" va se retrouver répliquée des dizaines de fois. Or, tout cela, il va bien falloir le stocker quelque part sous forme de bits et d'octets, et des informations inutilement répétées, ce sont des octets inutilement occupés… On n'en meurt pas, direz-vous. Certes, encore que.
2. Vu la manière dont nous avons bâti nos informations, nous ne sommes pas à l'abri d'une erreur, ou même d'un simple manque d'homogénéité, dans la saisie. Par exemple, lorsque nous avons tapé le genre du CD Foxtrot, nous avons oublié les deux "s" de "progressif", ce qui est une faute de frappe assez classique. De même, le genre "classique" est orthographié tantôt avec une majuscule, tantôt sans majuscule. Tout cela est fort préjudiciable pour la suite des événements. Si nous faisons une

recherche, par exemple, sur le genre "rock progressif" ou "classique", il manquera des CDs qui auraient dû y figurer. Cette organisation laisse donc la possibilité qu'existe une hétérogénéité des données, un des pires cauchemars de l'informaticien (qui pourtant n'en manque pas) .

Ces deux problèmes possèdent une solution commune, très simple mais extrêmement efficace, qui consiste à recenser séparément les CD et les genres:

TITRE	ARTISTE	ANNEE
Nursery Cryme	Genesis	1972
Foxtrot	Genesis	1972
Selling England by the Pound	Genesis	1973
Symphonie n°2	Jean Sibelius	1985
Symphonie n°7	Jean Sibelius	1987
Concerto pour violon	Felix Mendelssohn	1992
Crime passionnel	Jean Guidoni	1982
5th Gear	Brad Paisley	2007
Thick as a Brick	Jethro Tull	1973

GENRE
chanson française
classique
country
rock progressif

Il ne reste plus à présent qu'à préciser à quel genre appartient chaque CD. Pour ce faire, on identifie chaque genre par un code unique, et on reporter ce code dans un tableau.

CODE	GENRE
1	chanson française
2	classique
3	country
4	rock progressif

Avec:

TITRE	ARTISTE	ANNEE	CODE
Nursery Cryme	Genesis	1972	4
Foxtrot	Genesis	1972	4
Selling England by the Pound	Genesis	1973	4
Symphonie n°2	Jean Sibelius	1985	2
Symphonie n°7	Jean Sibelius	1987	2
Concerto pour violon	Felix Mendelssohn	1992	2
Crime passionnel	Jean Guidoni	1982	1
5th Gear	Brad Paisley	2007	3
Thick as a Brick	Jethro Tull	1973	4

Nous venons de faire d'une pierre deux coups:

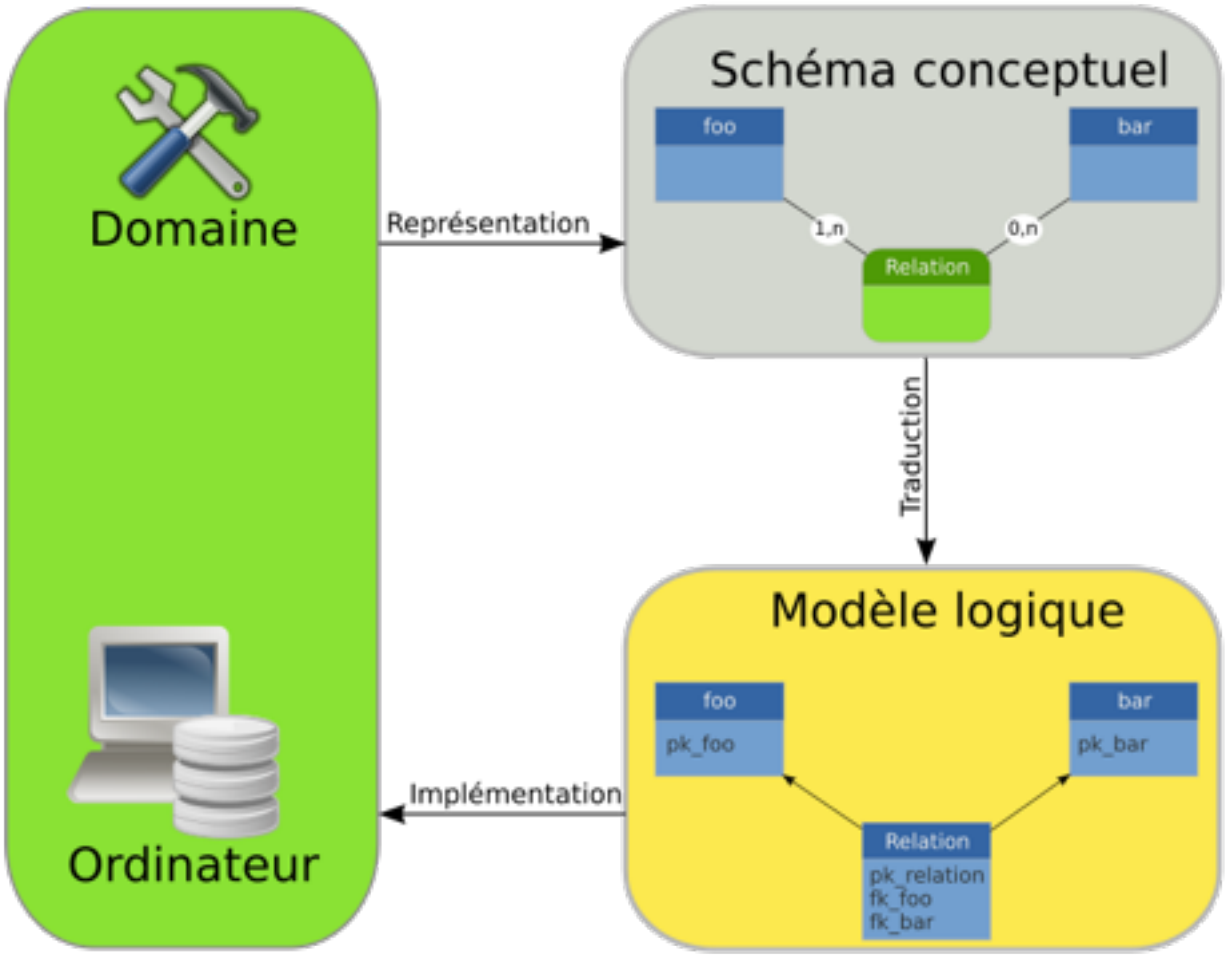
- on a économisé de la place en mémoire, car stocker un code mobilise beaucoup moins d'octets que stocker un intitulé
- surtout, on a pris une garantie contre les informations hétérogènes: le code correspond toujours au même intitulé, et on ne peut plus guère imaginer se retrouver avec un même genre musical orthographié différemment.

Ce que nous venons de faire, c'est le fond de la question en matière de modélisation de l'information: nous venons de créer une relation entre deux tables ; voilà pourquoi on parle de Systèmes de Bases de Données Relationnelles. Tout ce qu'on va voir ensuite, ce ne sont pour ainsi dire que des complications et des raffinements à partir de cette base simple.

Cela dit, jusqu'à maintenant, nous avons procédé pour ainsi dire uniquement par intuition – or, vous vous en doutez bien, il existe un certain nombre de formes pré-établies, de standards (tant de méthode que de représentation) pour venir à bout des problèmes les plus complexes. Ce sont ces formes et ces standards que nous allons aborder à présent.

Remarque: Il suffit d'observer la situation à laquelle nous sommes parvenus pour constater que nous n'avons parcouru que la moitié du chemin. Nous avons certes éliminé une source de redondances en créant la table Genres. Mais nous en avons laissé une deuxième: celle liée aux artistes. Il est donc essentiel de comprendre que ce qui est présenté ici n'est que le premier pas,

DIAGRAMME “ENTITÉS-RELATIONS”



Source: wikimedia.org

Nous utiliserons ici la méthode Merise 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).

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 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… heureusement nous avons normalement réalisé une modélisation des processus au préalable (voir premier chapitre).

Le MCD, les entités et les relations

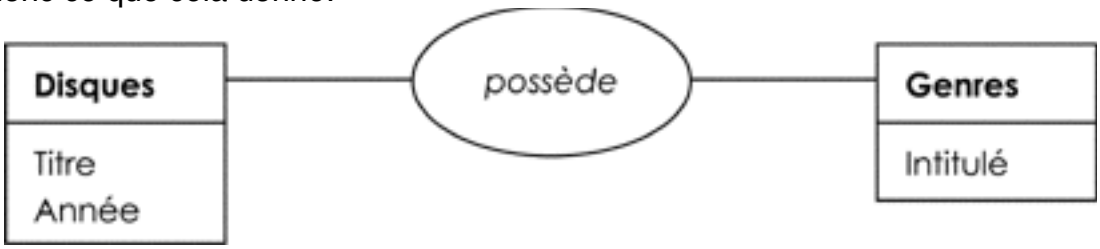
Les informations à traiter doivent être regroupées en ensembles cohérents, comme dans les tableaux que nous avons constitués il y a un instant. Dans les conventions de Merise, ces ensembles s'appellent des entités, et sont symbolisés par des rectangles. Chaque entité porte un nom, qui l'identifie de 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.

Éléments supplémentaires

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.
- Dans une table, les différents éléments individuels correspondent aux lignes, et sont appelés enregistrements.

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:

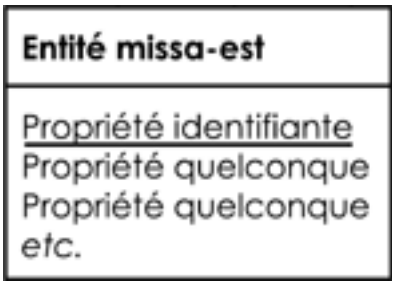
- Numérique: on y distingue systématiquement l'entier ("Integer") du nombre à virgule ("Float"). Le type "AutoIncrement", souvent utilisé pour gérer les clés primaires [MODE NO PANIC ON] vous saurez ce que c'est très bientôt [MODE NO PANIC OFF], correspond à un entier dont la valeur est automatiquement 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.

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 nous ne parlons 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:

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.

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'à chaque genre musical peuvent correspondre plusieurs disques. En sens inverse, en revanche, on peut être embêté pour décider combien de genres peut posséder chaque disque. On peut en effet très bien 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'utilisateur de la base de données. Il n'existe donc aucune règle qui permette de trancher 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 ces deux possibilités sur notre base de données.

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 va correspondre, au minimum et au maximum.

Ainsi, toute relation entre deux entités donnera lieu à la spécification de 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. Si nous poursuivons sur l'exemple de la discothèque, 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: pourrions nous avoir plusieurs CD du même genre, ou est-ce interdit ?)

Les valeurs possibles

Les cardinalités obéissent à un formalisme assez étroit:

- **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).
- **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).

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:

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 for-

ment 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.

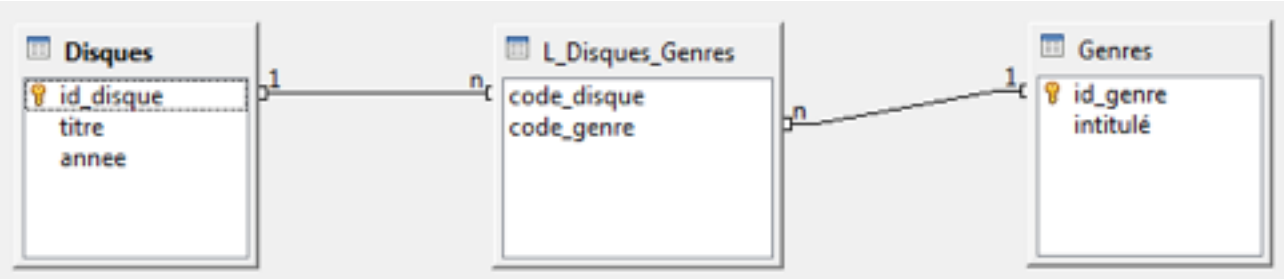
Du MCD au MLD: première approche

Rappelons 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. J Rappelons 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.

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:



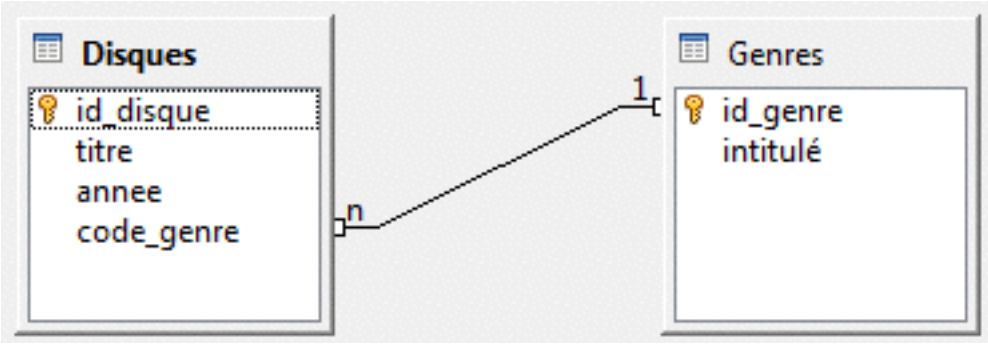
C'est très logique: la table de jonction va permettre d'associer tout élément de Disques à tout élément de Genres autant de fois que souhaité. Un même disque pourra ainsi être mis en rapport avec plusieurs genres, et un même genre avec plusieurs disques.

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)
2. 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:



3. Quand on y réfléchit, c'est parfaitement logique: pour chaque disque, il faudra renseigner un code (et un seul) qui devra correspondre à l'un des genres présents dans la table Genres. La table Genre, elle, ne contient aucun code renvoyant vers les disques, ce qui permet à chaque genre de pouvoir être en correspondance avec un nombre quelconque de disques. On vient de retrouver là 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.
4. On a une cardinalité (0, 1)
5. Cela correspond à la situation où chaque CD peut avoir un genre (au maximum) mais où il n'est même pas obligé d'en avoir un.
   Là, les informaticiens se divisent en deux catégories:
   Il y a les coulants (ou les laxistes, diront les autres). Ceux-là 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 (ou 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 si c'est une situation normale. Et puis, quand on va faire des recherches ou des traitements automatisés, cela risque de nous jouer de bien vilains tours. Ce n'est pas bien compliqué 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 - mais il faudra mettre en place un contrôle, en revanche, 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.

LA NORMALISATION DU MODÈLE

Ce vocable barbare recouvre l'idée que pour être valide, un modèle doit obéir à un certain nombre de normes. Celles-ci ne sont pas toutes du même ordre: certaines sont de simples règles de bonne conduite (ou d'hygiène), destinées à évacuer certaines ambiguités pouvant s'avérer gênantes à l'usage. D'autres sont impératives et leur violation constitue à coup sûr une faute grave. Le mieux est de les respecter toutes, en comprenant pourquoi il ne s'agit pas de préceptes tombés du ciel dans le seul but de nous pourrir l'existence, mais au contraire, de pratiques rationnelles visant à nous la simplifier.

Les redondances

C'est la règle principale, sur laquelle on ne saurait que trop insister: une modélisation correcte doit avoir éliminé toutes les redondances possibles de données en constituant des nouvelles entités chaque fois que nécessaire, comme nous l'avons fait dès le départ en regroupant les genres musicaux dans une entité séparée.

On doit donc explorer systématiquement toutes les propriétés de toutes les entités, en se demandant à chaque fois si les valeurs que prendront ces propriétés ne risquent pas de se répéter plus souvent qu'à leur tour.

Les valeurs multiples

Créer une propriété aux valeurs multiples constitue sans doute une des plus grosses fautes possibles dans une base de données.

Les valeurs multiples ne doivent pas être confondues avec les redondances. Par valeurs multiples, on entend le fait qu'une même propriété, pour chaque enregistrement, soit censée posséder plusieurs valeurs à la fois. Prenons l'exemple d'un journal qui souhaite mémoriser ses articles et leurs auteurs. On imagine donc une entité Articles, avec le titre, le corps de l'article, sa date, etc. ... Ayant bien assimilé la partie précédente, on prend soin de constituer une entité Auteurs et d'établir une relation entre Articles et Auteurs.

Fort bien. Sauf qu'à ce moment-là, le rédacteur en chef nous fait remarquer qu'un article peut fort bien avoir été écrit par plusieurs auteurs. "Qu'à cela ne tienne !", répondons-nous bravement. Et de là, il y a trois solutions... dont deux qui mènent droit à la catastrophe.

1. Que nous n’osons à peine concevoir, consiste à oublier tout ce que nous avons vu jusque-là et à attribuer une propriété Auteurs (au pluriel !) à notre entité Articles. C'est évidemment du total portnawak. D'une part, cette propriété viendrait tout à la fois télescoper l'entité Auteurs et la relation que nous avions établie avec elle. D'autre part, on se demande bien ce que pourrait recouvrir une propriété figurant au pluriel, et possédant donc plusieurs valeurs à la fois pour chaque enregistrement de la table...
2. C'est celle-là que nous appelons "la faute des valeurs multiples". Elle consiste à créer autant de propriétés que de co-auteurs possibles, en les appelant en général Auteur1, Auteur2, Auteur3, etc., et en considérant que chacune de ces propriétés est en relation avec l'entité Auteurs. Cette architecture va produire un galimatias à peine moins horrible que le précédent: outre qu'on nage en pleine confusion entre MCD et MLD, il est évidemment hors de question d'avoir une entité ressemblant à un emmental, avec des Null à tous les coins de rue (car tous les articles n'auront pas le maximum d'auteurs que nous avons prévu).

3. Ces deux fautes incarnent les deux formes possibles des propriétés à valeurs multiples. Elles sont tout aussi catastrophiques l'une que l'autre.
4. En fait, la troisième solution est non seulement la seule correcte, mais c'est aussi, en réalité, la plus simple car la relation entre Articles et Auteurs est de type plusieurs à plusieurs, et se traduira dans le MLD par la création d'une table de jonction.

Les synonymes

Les redondances d'informations au sein d'une même propriété ne sont pas les seules à devoir être traquées. Dans un modèle un peu compliqué, on peut vite être amené, volontairement ou non, à faire figurer plusieurs fois (dans plusieurs entités différentes) la même information – mettons, le nom du titulaire d'une police d'assurance. Lorsqu'une telle situation se produit, c'est toujours – hormis dans des contextes très particuliers – une erreur.

On se fixera donc comme principe qu'une information doit toujours figurer de manière unique dans un modèle. En effet, outre la perte de place occasionnée par une telle répétition, c'est surtout le risque d'une divergence entre les différentes copies de l'information qui est à craindre ; par exemple, une même personne qui se retrouverait au bout d'un moment répertoriée sous des noms à l'orthographe diverse selon qu'on est en train de parler de son assurance-vie, de son logement ou de ses versements de mensualités…

Ce deuxième principe peut être d'autant plus difficile à appliquer (mais d'autant plus nécessaire) que l'entreprise ou l'organisation elle-même répertorie les mêmes informations sous des noms différents (par exemple, le "code produit" dans un service, et la "référence" dans un autre). L'informaticien peut donc être trompé par ces noms différents et ne pas comprendre qu'ils désignent en fait la même information.

Ce type de biais est appelé, dans le jargon de Merise, un synonyme. Et les synonymes doivent donc être éliminés méthodiquement et sans pitié.

La polysémie

La polysémie pourrait être un nom de maladie. Dans un sens, c'en est d'ailleurs bien une. Elle est l'exact inverse du synonyme, mais elle est tout aussi nuisible à la bonne santé du système d’information.

Si le synonyme consiste à désigner la même information par des noms différents, la polysémie consiste à désigner des informations différentes sous le même nom. Par exemple, une propriété "référence" qui représenterait ici le code d'un produit, là celui d'un fournisseur, et ailleurs celui d'un client.

Même si dans l'absolu, la polysémie n'entraîne pas obligatoirement des problèmes fatals (tant que les polysèmes figurent dans des entités différentes), elle représente tout de même un choix bien dangereux. C'est une source potentielle de confusions. Or, en informatique, on a déjà bien assez de travail avec les problèmes inévitables ; les problèmes évitables, autant... les éviter.

Pour lutter contre la polysémie, il existe un truc aussi simple qu'imparable: utiliser systématiquement des noms de propriétés composés, qui se termineront par le nom de la table à laquelle elles appartiennent. En quelque sorte, le nom de la propriété est un prénom, celui de la table un nom de famille ! Ainsi, par exemple, on peut avoir comme propriétés code_disque, titre_disque, artiste_disque, etc. Dans le cas que nous évoquions à l'instant,

on ne risquerait plus de confondre référence_produit, référence_fournisseur et référence_client.

À pratiquer systématiquement donc, dès que la base commence à prendre un peu d'ampleur et que le risque de polysémie commence à montrer le bout de son nez.

TECHNIQUES AVANCÉES

Relations multiples

Il est tout à fait possible que deux entités soient reliées à la fois par plus d'une relation. Une telle possibilité n'a rien d'absurde, et correspond à beaucoup de situations de la vie réelle.
Imaginons, par exemple, que nous ayons à modéliser les relations entre une population et un parc immobilier. Nous aurons deux entités: l'une qui identifiera les différents individus (et dont les propriétés seront leurs noms, prénoms, n° de sécurité sociale, etc.). L'autre entité recensera les différents logements (n° d'appartement, étage, adresse, etc.). Or, il est clair que la relation d'un individu avec un logement peut être de plusieurs natures différentes, qui sont très largement indépendantes les unes des autres, il peut

- en être propriétaire
- en être locataire
- en être bailleur
- l'occuper à titre de résidence principale
- etc.

Face à ce problème, la solution est simple: les deux entités Individus et Logements doivent être reliées simultanément par plusieurs relations différentes. On parle alors d'associations plurielles.

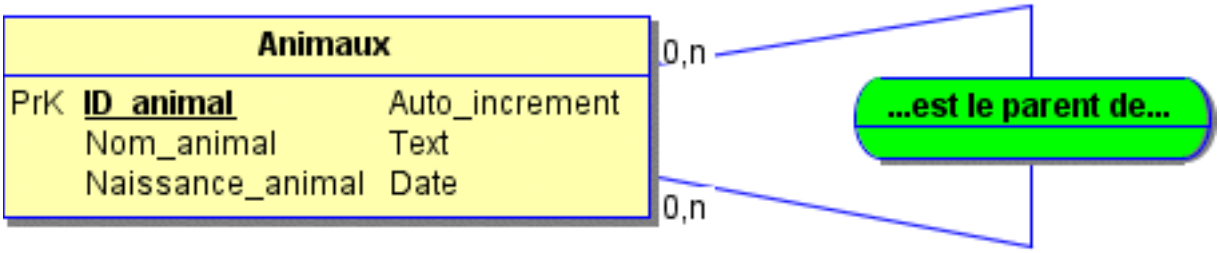
La réflexivité

Principe général

Encore un mot de matheux pour désigner une chose pas si compliquée qu'elle en a l'air, à savoir qu'une relation peut concerner des éléments d'une même entité.

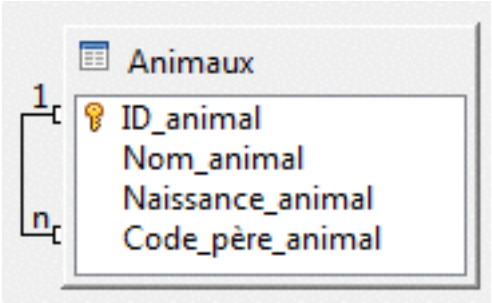
Supposons une base d'individus, par exemple les animaux d'un zoo. En vue de suivre les pedigrees de chaque bébête, nous devons enregistrer qui est l'enfant de qui. Pour cela, il nous faudra relier l'entité Animaux à elle-même, par la relation "est le parent de…". Cela peut paraître un peu bizarre, mais en réalité, cela ne pose aucun problème.

On pourra donc modéliser tout cela dans notre MCD de la manière suivante:

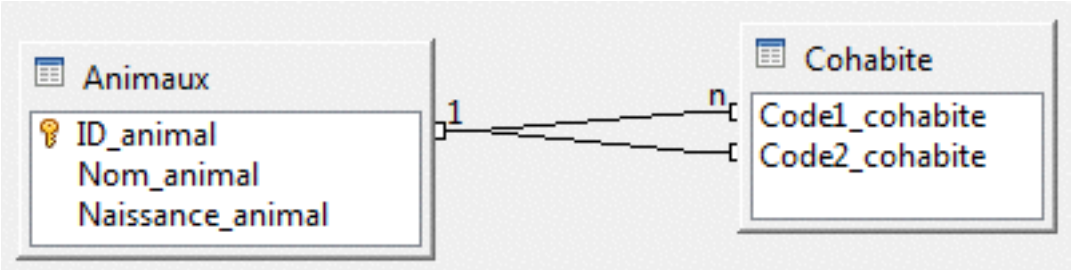


Du MCD au MLD

Au niveau du MLD, il n'y a rien de particulier à signaler – les règles exposées plus haut s'appliquent. Si la relation réflexive est de type "un à plusieurs", avec une cardinalité minimale de 1, cette relation prendra la forme d'une propriété supplémentaire dans la table, destinée à contenir l'ID de l'élément concerné. Par exemple, la relation "est le père de", qui entre dans ce cas de figure (un père peut avoir plusieurs enfants, mais un enfant ne peut avoir qu'un seul père) se manifestera par un champ code_père, dans lequel on entrera, le cas échéant, l'ID du père:



Si, en revanche, la relation réflexive est de type "plusieurs à plusieurs", elle donnera lieu à une nouvelle table. Celle-ci comportera deux champs, chacun des deux contenant l'ID d'un des deux animaux susceptibles de cohabiter. Et, toujours aussi logiquement, cette table de jointure sera reliée au champ ID_animal de la table Animaux par un double lien.



NB 1: on peut très bien imaginer une situation où une relation serait à la fois réflexive et plurielle ! En plus de savoir qui est le parent de qui, on pourrait ainsi noter qui peut être mis dans la même cage que qui (certains individus ne peuvent pas se blairer, et se battent dès qu'ils en ont l'occasion). Dans ce cas, pas de problème, l'entité Animaux sera greffée de deux relations réflexives différentes ("est le parent de" et "peut cohabiter avec").

NB 2: il serait en revanche tout à fait mal venu de créer deux relations, l'une "est le parent de" et l'autre "est l'enfant de". Ce serait en quelque sorte un pléonasme, ces deux relations n'en faisant en réalité qu'une seule.

Relations avec attributs

Principe général

Prenons le cas tout bête d'une entreprise qui gère les commandes de ses clients. Dans une première entité figurent les commandes, avec leur date, leur référence, etc. Pour savoir de quoi est composée chaque commande, on se tourne naturellement du côté d'une entité qui regroupe l'ensemble des produits vendus par notre entreprise. Au premier abord, il semble que nous soyons face à une situation déjà rencontrée maintes fois:

Sauf qu'il y a un petit souci: nous n'avons fait figurer nulle part la quantité dans laquelle chaque produit entre dans chaque commande. Par exemple, la commande n°456 peut porter sur 45 rideaux de douche à fleurs roses et 12 gants de toilette en crin, la commande n°508 sur 23 gants de toilette en crin et 3 savons de Marseille, etc. Or ces nombres (45, 12 et 23) doivent obligatoirement être stockés quelque part, faute de quoi mon système de gestion de factures ressemblera à une passoire.

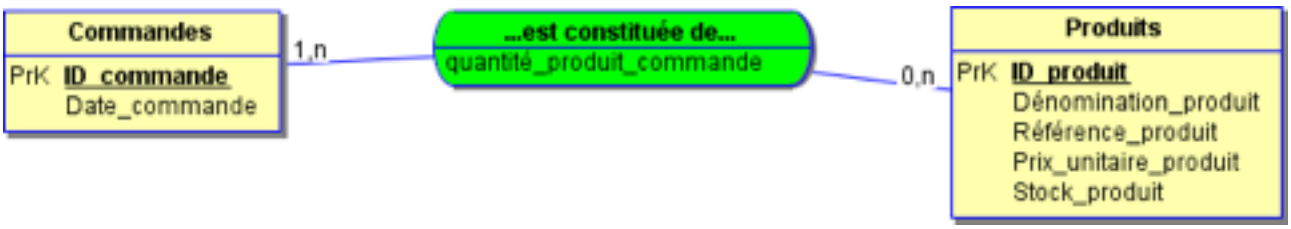
Première possibilité: nous les intégrons à la table des commandes. Hum... Comment faire ? En créant un seul attribut ? Mais cela veut dire que pour une commande donnée, on ne pourra renseigner qu'un seul nombre. Or, dans une commande, on peut commander plusieurs produits çà la fois (chacun dans une quantité différente). Donc, ça ne va pas.

Deuxième possibilité: nous intégrons les quantités à la table des produits. Mais là, c'est le même problème: Si nous créons un nouvel attribut (et un seul), cela veut dire qu'à chaque produit correspond une quantité et une seule. Sauf qu'évidemment, Il se peut très bien que le nombre de Kiki en peluche commandés par Bidule soit de 7 546 tandis que Machin, lui, en a acheté 12. Donc, problème.

On n'ose même pas imaginer créer plusieurs attributs (quantité 1, quantité 2, quantité 3, etc.): que ce soit dans une table ou dans un autre, ce serait un flagrant délit de propriété à valeurs multiples, dont on a vu précédemment qu'il s'agissait d'une faute irrémédiable.

Nous voilà donc bien avancés... En fait, toute la difficulté vient du fait que le nombre de produits de chaque sorte qui ont été commandés est une caractéristique liée à chaque commande. La solution s'impose d'elle-même: cette information doit donc figurer au cœur même de la relation qui unit les commandes aux produits. À chaque fois qu'une nouvelle commande porte sur un nouveau produit, on doit se demander: “en combien d'exemplaires ?”.

L'information (la propriété) "Quantité commandée" n'est donc pas un attribut d'une des deux entités Commandes et Produits, mais comme un attribut de la relation qui les unit. Ce que nous représenterons de la manière suivante:



Remarque vraiment intelligente: une relation ne peut posséder légitimement un attribut que si elle est de type "plusieurs à plusieurs". Démonstration par l'absurde:

- Imaginons qu'il ne puisse y avoir qu'un seul produit par commande. Dans ce cas, la quantité commandée serait unique pour chaque commande, et pourrait donc être un attribut de l'entité Commande.
- Inversement, si chaque produit ne pouvait être commandé qu'une fois et une seule, on pourrait ajouter la quantité comme attribut de l'entité Produits

Pour conclure, ajoutons que rien n'oblige une relation à posséder un seul attribut. Il est tout à fait possible qu'existent des relations avec deux, trois, quatre, etc. attributs.

Passage au MLD

Aucune difficulté: une relation avec attributs étant nécessairement une relation "plusieurs à plusieurs", elle donnera naissance, lors du passage au MLD, à une nouvelle table (de jointure). Tout simplement, les attributs de la relation deviendront des propriétés supplémentaires de la table de jointure (en plus des codes correspondant aux clés primaires des tables concernées).

Relations de dimension supérieure à 2

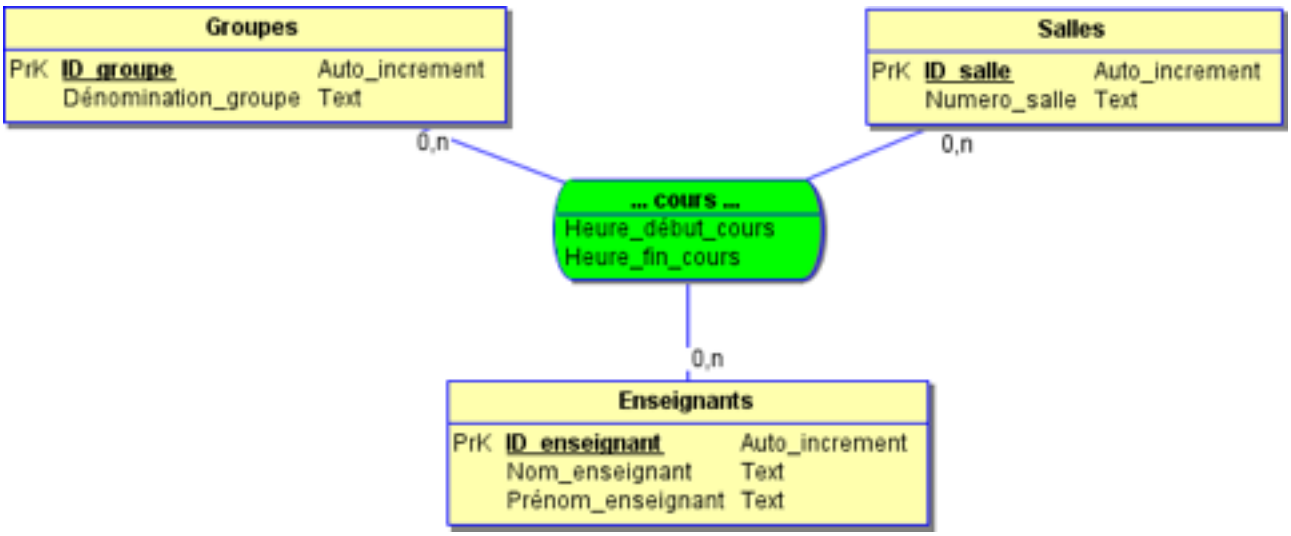
Aïe aïe aïe, encore du vocable matheux. Mais non, c'est rien, ça pique à peine au début et après on ne sent plus rien.

Principe général

Partons simplement du fait que jusqu'à présent, les relations que nous avons modélisées ne mettaient toujours aux prises que des entités deux à deux: les CD et les genres musicaux, les commandes et les produits, etc. Il peut cependant arriver que la relation doive s'établir directement entre plus de deux entités (d'où le titre).

Prenons un système d'information qui gère le travail quotidien dans une université (bon courage). Chaque jour, des groupes d'étudiants ont cours dans certaines salles avec certains professeurs. Mais on suppose que tout dans cette histoire est susceptible de varier: dans une journée donnée, le même groupe peut avoir cours avec plusieurs profs dans plusieurs salles différentes, les profs peuvent donner plusieurs cours ou aucun, etc.

La manière la plus directe de modéliser une telle situation sera d'écrire que:



On est donc face à une relation impliquant non plus deux, mais trois entités, ce qui finalement, ne pose pas de problèmes particuliers. Au passage, on remarque qu'une relation de dimension supérieure à 2 peut fort bien comporter des attributs, comme toute autre relation (Ici, Heure_début_cours et Heure_fin_cours).

Du MCD au MLD

On vérifie facilement qu'une relation de dimension supérieure à deux est forcément de type "plusieurs à plusieurs" (faudrait-il dire: "plusieurs à plusieurs à plusieurs" ? – en voilà une question qu'elle est bonne). Une telle relation devient donc systématiquement une table de jointure, comportant autant de champs que d'entités à relier.

Allez, là, ça va carrément être l'extase

SITUATIONS EMBARRASSANTES

Un des problèmes les plus fréquemment rencontrés par le modélisateur débutant est de savoir comment définir les relations entre plusieurs entités qui "marchent ensemble" - mais comment ? Si l'on prend le cas où l'on a trois entités, il y a trois grandes façons de les relier:

Deux relations simples (figure A)

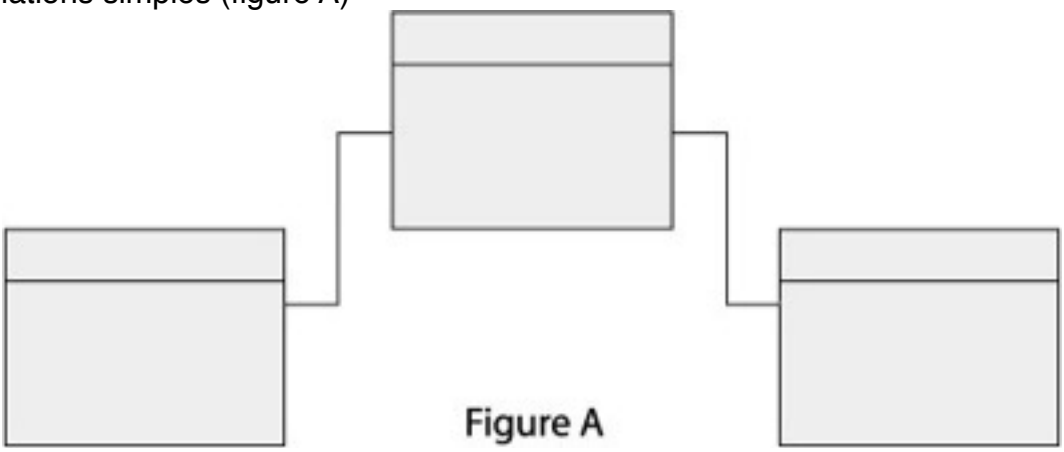


Figure A

Trois relations, reliant les trois entités deux à deux (figure B)

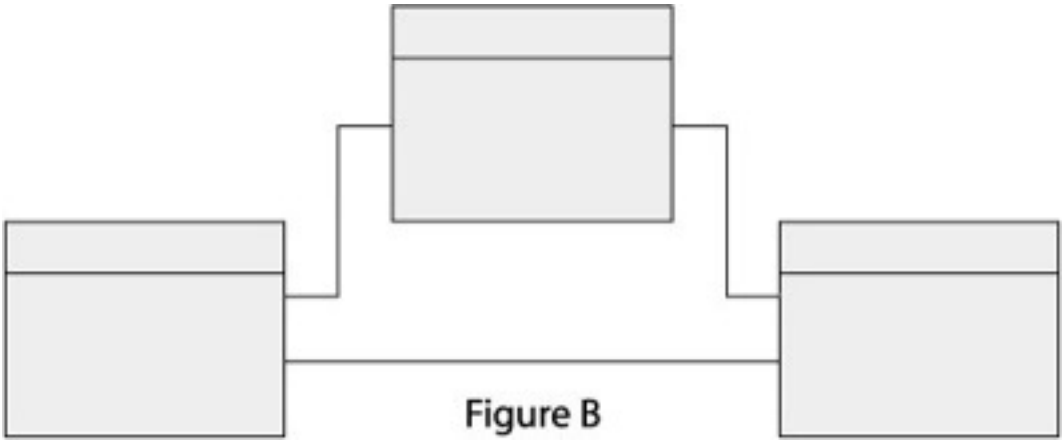


Figure B

Une seule relation de dimension trois (figure C)

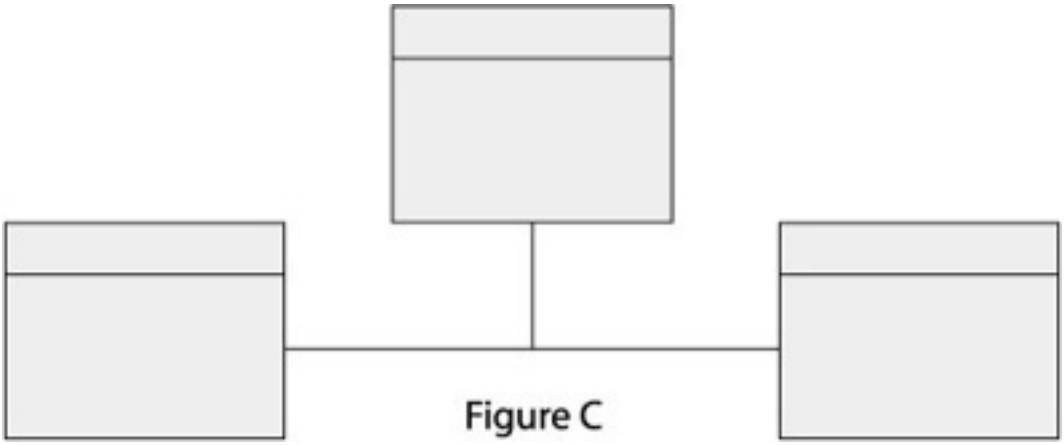


Figure C

Arrivés à ce point, nous avons pour vous deux mauvaises nouvelles et une bonne.

Première mauvaise nouvelle: les trois solutions ne sont jamais équivalentes. Autrement dit, opter pour une des trois solutions au pif vous conduira dans le décor deux fois sur trois en moyenne (si en plus vous êtes malchanceux, ça peut être plus souvent).

Deuxième mauvaise nouvelle: il n'y a pas de truc miracle qui permette de choisir la bonne option sans avoir à réfléchir au problème à résoudre. Autrement dit, tout réflexe du genre "J'ai fait comme ça la dernière fois, ça a marché" ou "Ca m'a l'air sympa ce joli dessin-là" a peu de chances de vous conduire au bon résultat.

Maintenant, la bonne nouvelle: la modélisation étant une activité rationnelle, elle ne relève pas des processus cognitifs de la magie noire. Donc, si on réfléchit avec un chouia de méthode, on trouve la bonne solution.

Quelques remarques générales

Dans la figure A: comme on le constate, il n'existe pas de lien direct entre les éléments de l'entité 1 et ceux de l'entité 3. Ce lien est établi de manière indirecte (par transitivité, diraient les matheux) mais il faut raisonner à partir des cardinalités pour vérifier si les infor-

mations peuvent bel et bien être retrouvées de la manière dont l'exigent les besoins du problème.

Dans la figure B: le fait majeur est que chaque entité est doublement reliée aux deux autres. La table 1 est reliée directement à la table 3, et elle lui est aussi reliée via l'entité 2. A partir de là, de deux choses l'une.

- Soit ces relations expriment toutes une même réalité: par exemple, si l'on prend des clients, des commandes et des produits, on dira que les clients passent des commandes, que les commandes portent sur des produits, et que les clients commandent des produits. Il y a là un problème d'architecture, car la même réalité est codée deux fois. On s'expose alors à ce que l'une des deux voies de relation disent autre chose que l'autre... ce qui est catastrophique. Et si elles disent toutes les deux la même chose, c'est bien qu'il y en a une qui ne sert à rien...
- Soit ces relations ne parlent pas toutes de la même chose. Par exemple, la relation entre Clients et produits ne porte pas sur des affaires de commandes, mais dit que tel client a droit à une remise sur tel produit. Dans ce cas, pas de souci: la relation peut fort bien paraître doublée, en réalité, elle ne l'est pas: il s'agit bien de deux chemins de relations différents.

Dans la figure C: les trois entités sont associées "simultanément". Une manière plus juste de s'exprimer sera de dire que tout élément de chaque entité peut librement être associé à tout élément des deux autres entités. Ce modèle exprime la plus grande souplesse, puisque toutes les combinaisons sont possibles. Il s'applique aux situation dans lesquelles il n'y a pas de régularité (certains éléments étant associés à certains autres, et seulement à ceux-là).

Brève étude de cas

Les raisonnements généraux valent ce qu'ils valent, mais pour mieux se retrouver dans tout cela, il n'est pas inutile de partir d'un exemple. Imaginons que nous modélisions un centre de formation. Nous aurons donc à gérer (entre autres) des profs, des groupes d'élèves et des salles d'enseignement. On s'en doute, il existera une entité profs, une entité groupes et une entité salles. Mais la difficulté est de déterminer quelle sera la nature des relations que ces entités devront entretenir entre elles.

Hypothèse n°1: chaque prof a en charge un certain nombre de groupes (toujours les mêmes sur une période donnée), et chaque groupe a toujours cours dans la même salle Il faut bien évidemment relier les profs aux groupes (chaque groupe a-t-il un seul prof ou peut-il en avoir plusieurs, on ne le sait pas a priori, mais cela ne change rien pour la suite de notre raisonnement). De même, on doit relier les groupes aux salles. On a donc a priori une figure de type A, avec comme entité centrale les groupes. On peut néanmoins se demander s'il ne faudrait pas ajouter à cela une relation directe entre les profs et les salles, comme dans la figure B, pour dire "tel prof a cours dans telle salle". On comprend vite que cela ne servirait qu'à compliquer les choses, sans apporter la moindre information supplémentaire. En effet, avec une architecture de type A, on sait déjà que tel prof a en charge telle groupe, et on sait aussi où chaque groupe a cours. On peut donc facilement retrouver dans quelles classes enseigne chaque prof. Si nous adoptons la figure B, la relation supplémentaire qu'elle introduit signifie qu'on va dorénavant noter ce renseignement indépendamment du cours concerné. Soit cette information supplémentaire est cohérente avec celle que me donnait la chaîne qui reliait les profs aux classes et les classes aux

cours, et elle n'apporte donc aucune information supplémentaire. Soit elle n'est pas cohérente avec ce qu'on peut reconstituer au travers de la chaîne profs - classes - salles, et l'administrateur de la base de données va avoir quelques nuits blanches. Bref, dans les deux cas, ce ne sont que des ennuis et aucun avantage. Reste la possibilité d'une seule relation tridimensionnelle (figure C). Il faudrait alors informer le système de chaque combinaison prof-classe-salle. A priori, tout va bien... sauf que nous ouvririons ainsi la possibilité de saisir que le groupe X a cours tantôt en salle 120, tantôt en salle 124, etc. Or, il était bien stipulé que chaque classe avait toujours cours dans la même salle. Exit donc cette solution.

Hypothèse n°2: au coup par coup, un prof prend en charge un groupe et fait cours dans une salle. Les salles ne sont pas toujours attribuées aux mêmes profs ni aux mêmes groupes. Et les groupes ne sont pas toujours pris en charge par les mêmes profs. On reconnaît là la situation où chaque événement rassemble à la fois un prof, un groupe classe et une salle, sans aucune contrainte ni obligation de régularité. Autrement dit, il faut que mon modèle autorise à associer à tout moment tout élément des entités profs, groupes et salles: il s'agit évidemment d'une relation du type de la figure C. Par acquit de conscience, regardons ce qui se passerait si nous opterions pour la figure A. Dans ce cas, nous pourrions établir les relations entre profs et groupes et entre groupes et salles. Mais, du fait que chaque prof serait potentiellement associé à plusieurs groupes, chaque groupe à plusieurs profs et à plusieurs salles, et chaque salle à plusieurs groupes, nous n’aurions aucun moyen de reconstituer, à un moment donné, quel prof a cours dans quelle salle. Rajouter une relation directe entre les profs et les salles, avec une figure de type B, ne m'avancerait pas à grand chose: nous saurions quels profs ont cours dans quelles salles, mais pas avec quelle classe (de même, nous continuerions à savoir quels groupes ont cours dans quelle salle, mais pas avec quel prof, et quels profs ont cours avec quel groupe mais pas dans quelle salle).

Hypothèse n°3: les profs ont en charge certains groupes, qui ont toujours cours dans les mêmes salles. Mais sur certains créneaux horaires, les salles servent aussi de lieux de permanence pour les profs.

Comme vous vous en doutez, voilà un cas qui doit être modélisé par la figure B. En effet, dans cette hypothèse, il existe entre les profs et les salles deux relations de nature tout à fait différente. D'une part, les profs peuvent accéder à une salle dans le cadre de leur permanence: c'est la relation directe entre les deux entités. D'autre part les profs sont affectés à des groupes qui sont eux-mêmes affectés à des salles d'enseignement: c'est la relation indirecte (des profs aux groupes, et des groupes aux salles).

Opter pour la figure A serait s'interdire de pouvoir enregistrer les permanences. Opter pour la figure C serait de surcroît ouvrir la porte au fait que chaque classe ne serait pas toujours dans la même salle.

Conclusion: Répétons-le, des trois figures présentées ci-dessus, aucune n'est "juste" ou "fausse" en elle-même. Toutes sont des modélisations pertinentes de certaines situations réelles... et inadaptées à d'autres situations. Nous ne pouvons donc que répéter: il n'y a pas de "truc" qui dispense de réfléchir à chaque problème pour trouver la meilleure solution...

Sources et bibliographie

- *D. Torres & autres, “Bases de Données”, TECFA (http://ecol2.com/u/0k0my2)*
- *C. Darmangeat, “Petit cours de Modélisation”, U. Paris 7 (http://ecol2.com/u/shngbz)*
- *J. Carrière, “Introduction au BPMN 2.0”, U. du Québec à Montréal (http://ecol2.com/u/qtq2fp)*