Exercices

Exercice 1 Notion de base de données

Énoncé

Quelles sont les différences majeures entre un fichier informatique et une base de données gérée par un SGBD ?

Solution

On peut lister ces points essentiels qui les différencient :

- Il n'est pas nécessaire de connaître la méthode de stockage des informations sur le disque pour manipuler les données avec une base de données.
- Un fichier informatique simple n'est pas conçu pour effectuer une recherche d'information par le contenu : pour retrouver le(s) enregistrement(s), on est obligé de parcourir tout le fichier.
- Les modifications de structure (ajout/suppression d'un champ ou modification de sa taille…) nécessitent de recréer un autre fichier et d'y recopier les données.
- Une base de données contient en général plusieurs fichiers dont les enregistrements sont reliés entre eux.

Exercice 2 Recherche dichotomique

Énoncé

Donnez un algorithme intuitif simple de recherche dichotomique en utilisant une table d'index et une table à accès direct.

Solution

Soit V la valeur recherchée, Ti le tableau d'index de taille n. On suppose que la table d'index contient les valeurs du champ indexées dans sa colonne 1 et les numéros d'enregistrement correspondants dans sa colonne 2.

Si le tableau est réduit à un élément dont la valeur dans la première colonne Ti[1,1] est différente de la valeur recherchée alors la recherche est un échec sinon comparer l'élément du milieu z du tableau Ti avec la valeur V

Si l'élément Ti[z,1] est égal à la valeur, accéder à l'enregistrement directement par son numéro Ti[z,2]

Si l'élément Ti[z,1] est inférieur à la valeur recommencer avec la partie basse du tableau (de 1 à z-1)

Si l'élément Ti[z,1] est supérieur à la valeur recommencer avec la partie basse du tableau (de z+1 à n)

EXERCICE 3 LANGAGES D'UN SGBD

Énoncé

On veut supprimer tous les enregistrements qui contiennent la valeur 666 dans le champ 'catégorie'. Utilise-t-on le langage de description de données ou le langage de manipulation de données ? Que se passerait-il si l'on voulait augmenter la taille du champ 'catégorie'.

Solution

Le langage de description de données s'intéresse à la modification de structure d'une table déjà créée ou à la gestion des tables (création/modification). Dans notre cas, on ne touche pas à la structure, on supprime des enregistrements, donc des données de la table. On ne touche pas au dictionnaire de données. On utilisera par conséquent le langage de manipulation de données du SGBD. Pour augmenter la taille du champ, on modifie cette fois la structure même de la table, on utilise alors le langage de description de données.

EXERCICE 4 MODÈLES DE REPRÉSENTATION

Énoncé

Vous devez représenter l'organisation de données correspondant à une classification scientifique d'espèces d'oiseaux. Quel modèle de données (hiérarchique, réseau, relationnel, objet…) choisiriez-vous ?

Solution

Par nature, ce type de données est structuré strictement de manière arborescente et cette structure reste assez stable dans le temps. Il est donc tout à fait possible d'utiliser un simple modèle hiérarchique. Un modèle réseau ne sera pas utile en principe du fait de la structure arborescente des données. On peut également utiliser les modèles relationnel ou objet, mais il n'apporteront pas d'avantage décisif dans ce cas (très) particulier.

EXERCICE 5 MÉTIERS DES BASES DE DONNÉES

Énoncé

Est-il possible de faire réaliser toutes les étapes de la conception d'une base de données par une même personne ? Si oui, quelles sont alors ses compétences minimales ?

Solution

Dans une petite structure, c'est souvent la même personne qui réalise l'ensemble du processus de construction d'une base de données. Ce n'est évidemment pas la bonne méthode, car la vision d'un système d'information élaboré par un administrateur de base de données est très orientée par le SGBD qu'il emploiera. On peut facilement faire le parallèle avec le développement de logiciels où un programmeur va avoir une approche déformée par les préoccupations liées au langage plutôt que d'adopter un point de vue sur la structure générale de l'application. Au minimum, la personne devra disposer des compétences en conception de base de données et en administration du SGBD qui sera utilisé.

EXERCICE 6 UTILISATEURS D'UNE BASE DE DONNÉES

Énoncé

On utilise pour cet exercice la base de données exemple de vente de voitures. On considère les opérations suivantes :

1. Ajouter une personne dans le fichier client.
2. Modifier les possibilités d'ajout dans le champ 'couleur' .
3. Augmenter la taille du champ 'couleur'.
4. Sortir le chiffre d'affaires par marques pour le mois en cours afin de l'importer dans un tableur.
5. Enregistrer une vente.

Quels types d'utilisateurs sont concernés par ces opérations ?

Solution

1. Un utilisateur final (par exemple un vendeur) muni des droits appropriés peut intervenir sur le contenu des données.
2. Pour cette opération, cela dépend s'il s'agit d'une convention ou si cela est entré au niveau des contraintes du SGBD. Dans le premier cas, un utilisateur final peut s'en charger ; dans le second, il faut recourir au concepteur ou à l'administrateur de base de données.
3. Pas d'ambiguïté ici, car on touche à la structure même des données ; cela est du ressort du concepteur ou de l'administrateur de base de données.
4. Il s'agit du domaine du programmeur d'application qui récupère les données en utilisant le SGBD et qui les traite dans un programme pour leur donner leur forme finale.
5. Un utilisateur final (par exemple un vendeur ou la comptabilité) muni des droits appropriés peut intervenir sur le contenu des données.

EXERCICE 7 VUES EXTERNES

Énoncé

On utilise également pour cet exercice la base de données exemple de vente de voitures. On considère trois types d'utilisateurs de la base :

1. les clients ;
2. les vendeurs ;
3. le service comptabilité.

Quelles sont les vues à prévoir pour ces catégories d'utilisateurs ?

Solution

1. Les clients ne doivent avoir accès en lecture qu'aux informations concernant les voitures en stock (non encore vendues).
2. Les vendeurs, s'ils gèrent également le parc de voitures comme c'est souvent le cas, peuvent avoir accès en lecture et en écriture à toutes les données (ventes, voitures, personnes).
3. Le service comptabilité peut avoir accès en lecture à toutes les informations, mais ne peut modifier les informations concernant les voitures ou les personnes du fichier client.

EXERCICE 8 BASE DE DONNÉES RÉPARTIES

Énoncé

On décide de recopier régulièrement une base de données complète sur chacun des six sites de l'entreprise. Quel est l'intérêt de cette solution ?

Solution

C'est une solution coûteuse en ressources, en particulier pour la synchronisation de toutes les mises à jour, mais qui peut remplir deux fonctions :

1. De toute évidence, l'idée est que les différents sites de l'entreprise accèdent à leur copie locale des données. Cela permet d'accélérer les accès et de répartir la charge sur les serveurs locaux de chaque site. Accessoirement, la circulation des requêtes d'interrogation et de mise à jour peut être limitée au réseau local, ce qui renforce la sécurité.
2. Il existe six copies des données sur des sites géographiquement séparés. En cas de sinistre, on peut repartir sans problèmes avec une des copies de la base.

Avant de mettre en place un tel dispositif, on doit se poser la question de la mise à jour des données. Est-ce que les modifications se font uniquement sur la base « maître », ce qui semble plus raisonnable, ou peut-on les effectuer sur toutes les bases et « consolider » ensuite ?

EXERCICE 9 XML

Énoncé

Pourquoi préfère-t-on utiliser XML plutôt que HTML pour représenter les données provenant d'une base de données ? Les données XML sont dites « autodescriptives ». Qu'est-ce que cela signifie et par quel(s) dispositif(s) est-ce réalisé ?

Solution

Par nature, les données contenues dans une base de données sont structurées. Le langage HTML a été conçu pour décrire la mise en forme d'un texte sans considération de sa structure interne. Donc, il n'est pas adapté si l'on désire conserver la structuration des données. Le langage XML a été précisément créé pour décrire la structure des données. Il est toujours possible de passer ensuite du langage XML au langage HTML par une feuille de style ; l'inverse n'est pas possible.

XML permet de représenter des structures de données différentes sous forme arborescente ; il est donc nécessaire de posséder une description de la « grammaire » de la structure. Ce document accompagnateur d'un fichier XML est une DTD, comme pour les fichiers XML classiques, ou plus commodément un schéma XML. Un des avantages du schéma est que l'on peut utiliser les mêmes algorithmes de parcours que pour le fichier XML.

EXERCICE 10 FOUILLE DE DONNÉES ET ENTREPÔTS DE DONNÉES

Énoncé

Par des méthodes d'analyse d'associations, on découvre qu'en utilisant votre système d'information les clients dont le nom commence par M achètent le samedi plus de produits que les autres. Ils génèrent donc un chiffre d'affaires plus important, mais ces produits sont à marge faible et le bénéfice est moins important que pour ceux dont le nom commence par un Z. Qu'avez-vous intégré comme données dans votre entrepôt de données pour pouvoir effectuer cette opération en supposant que votre entreprise soit organisée avec une structure de services classique ?

Solution

Pour obtenir les éléments nécessaires à l'analyse, il nous faut intégrer les données de différents services de l'entreprise.

- Les données du fichier clientèle sont gérées par le service commercial (peut-être avec un tableur ou un traitement de texte pour faire des mailings) afin d'obtenir le nom des clients.
- Les données du service comptabilité permettent d'obtenir les journaux de ventes ; la gestion est faite par exemple par une application de gestion spécifique connectée aux caisses.
- Les données du service achat sur les négociations avec les fournisseurs peuvent être gérées par exemple par une application développée en interne. La marge provenant de la négociation avec les fournisseurs est fluctuante pour le même article au cours du temps en fonction du marché, des personnes qui négocient, etc. Il serait donc intéressant pour ces données de disposer des valeurs « historisées » par périodes pour affiner l'analyse par des tendances.

Pour intégrer ces données provenant de sources différentes dans votre entrepôt de données, vous utiliserez une (ou plusieurs) application(s) de type ETL (*Extract, Transform and Load*) que l'on appelle aussi « datapumping ».

Chapitre 2

Analyse du monde réel

Ce chapitre présente la première étape du processus de modélisation du monde réel, qui consiste à recueillir les informations puis à les transcrire sous une forme conduisant à un passage aisé au modèle relationnel.

On utilise à cette fin le modèle entité-association, dont les concepts et la mise en œuvre sont présentés dans ce chapitre. Cette démarche de modélisation est utilisée depuis plus de vingt ans et présente l'intérêt de proposer une méthode d'analyse simple et efficace dans la majorité des cas.

Au cours des dernières années, la représentation utilisant le formalisme du modèle entité-association est progressivement remplacée par le langage de modélisation UML. Ce dernier apporte, en plus des avantages de la normalisation, de la disponibilité d'outils graphiques logiciels ainsi que des possibilités étendues de description. Les bases de la notation UML sont abordées à la fin de ce chapitre.

1 Démarche d'analyse

1.1 Approche du monde réel

Comment appréhender et simplifier le monde réel, par nature complexe, pour réaliser une modélisation ?

Cette tâche relève de compétences multiples du domaine d'un cabinet de consulting. Il est nécessaire d'identifier les besoins des utilisateurs ainsi que les objectifs et les processus d'alimentation en données des systèmes d'information à concevoir. Les bases de données sont dorénavant au cœur de la plupart des applications, et leur structure doit correspondre au mieux aux attentes de l'organisation.

Son élaboration nécessite différentes étapes et se déroule souvent en même temps que le processus d'**analyse du problème**. Des allers-retours entre ces différentes étapes de la conception sont souvent nécessaires à la constitution du **modèle conceptuel** de la base. On qualifie alors ce processus d'itératif. Il s'agit de construire la structure de la base de données par raffinements successifs.

La première phase de l'analyse du monde réel du problème est réalisée par des **entretiens**, généralement codifiés, avec les utilisateurs. On effectue une analyse du discours pour en extraire l'information utile que l'on resitue dans le contexte de l'organisation en général. L'objectif principal est de guider l'**analyste**, on utilise des **méthodes d'analyse et de conception** issues de l'étude des flux d'information de l'entreprise. Parmi celles-ci, on peut citer la méthode **Merise** d'origine française – très répandue en France dans les années 1980 – ainsi que d'autres issues de la recherche et du génie logiciel, ou spécifiques à des grandes entreprises de consulting.

La présentation de ces méthodes fort complexes dépasse largement le cadre de cet ouvrage. L'objectif de cette section est de donner quelques pistes pour approcher la réalité à modéliser. L'expression des besoins repose sur la formulation du problème à l'aide de phrases simples qui décrivent la réalité à modéliser. Ces phrases se présentent sous la forme « sujet-verbe-complément », avec une tournure active quand cela est possible. Le but est d'obtenir deux types de phrases :

- Celles qui décrivent les liens entre les objets du monde réel – généralement une action ou une propriété. Exemple : *Un lecteur emprunte un livre. Un livre a un auteur.*
- Celles qui caractérisent la manière dont sont reliés ces objets. Exemple : *Un lecteur est considéré comme lecteur s'il a au moins déjà emprunté un livre. Un livre peut être emprunté par plusieurs lecteurs. Il n'y a pas de livres anonymes, un livre est écrit par au moins un auteur.*

On doit ensuite préciser les données qui constituent les objets ainsi que celles qui caractérisent les liens entre les objets.

Remarque

Le terme d'objet du monde réel employé ici n'est pas pris au sens de la programmation objet. Il s'agit plutôt de caractériser un regroupement logique de données. Un titre, un auteur, un éditeur constituent un livre. Un nom, un prénom, un numéro de Sécurité sociale constituent une personne.

1.2 Mise en œuvre

Comment procéder intuitivement pour obtenir ces phrases ?

Un bon point de départ consiste à faire l'inventaire des objets tangibles ou perçus du monde réel. Une fois ces objets identifiés, on cherche à exprimer le (ou les) lien(s) qui permet(tent) de les associer. Par exemple, si l'on doit modéliser une activité de location de DVD, les objets que l'on peut appréhender immédiatement sont les DVD et les clients. En ce qui concerne les liens entre ces objets, on note qu'un client *réserve* un DVD ou qu'un client *loue* un DVD, etc. Ensuite, il faut identifier les objets moins faciles à percevoir directement : les fournisseurs, les acteurs, les réalisateurs… Enfin, une fois les objets identifiés, on cherche à qualifier les liens trouvés. Il faut tenir compte du fait que le lien est toujours à double sens. Par exemple, un client *emprunte* plusieurs DVD. Un DVD *est emprunté* plusieurs fois ou *n'est jamais emprunté* par un client.

Pour obtenir ce résultat, quelles questions faut-il se poser et quelles questions doit-on poser aux acteurs de l'organisation ?

- Décrivez l'activité globalement, en termes simples, sans entrer dans les détails, pour identifier les objets et leurs liens éventuels.
- Indiquez quelles sont les « procédures » utilisées dans l'activité pour caractériser les liens entre les objets. Les procédures permettent d'énoncer les contraintes qui seront intégrées ensuite dans la base de données.

On note que l'on modélise souvent des actions qui représentent une activité, plus rarement des éléments statiques. Les actions représentent fréquemment le lien entre les objets : une personne *emprunte* un DVD, une voiture *est achetée* par un client, etc.

1.3 Notion de temps

Le temps est une notion importante, une base de données modélise des actions qui ont lieu durant une période de temps. Il faut toujours avoir à l'esprit cet aspect pour éviter des erreurs de conception. Une erreur classique est de confondre l'aspect simultané d'une action avec la possibilité de la réitérer durant la période concernée. Lorsque l'on spécifie qu'« un livre peut être emprunté plusieurs fois », il est évident qu'un livre ne peut être emprunté par deux personnes simultanément, mais plutôt qu'il pourra être emprunté à plusieurs reprises durant la période modélisée du fonctionnement de la bibliothèque.

1.4 Cas pratique

Afin d'illustrer les recommandations précédentes, on considère la modélisation très schématique du fonctionnement d'un hôtel. Quelle(s) phrase(s) simple(s) décri(ven)t l'activité de l'hôtel ?

On peut proposer en première approche cette phrase : « Un hôtel loue des chambres à des clients qui effectuent des réservations. » Après avoir procédé à l'analyse de la phrase pour en extraire les parties importantes, on peut la réécrire de la manière suivante :

Un client loue une chambre ; un client réserve une chambre.

Les deux **objets** du monde réel – la chambre et le client – apparaissent clairement. On a identifié ici un double lien entre ces deux objets, cas assez fréquent.

- Ensuite, on s'intéresse à la caractérisation des **liens** : Une chambre peut n'avoir jamais été louée ni réservée.
- Un client est inséré dans le système d'informations à partir du moment où il a effectué soit une réservation soit une location.
- Un client peut réserver ou louer plusieurs chambres.
- Une chambre peut être réservée ou louée plusieurs fois, mais pas pendant la même période de temps.

On rappelle que l'on considère toujours une modélisation associée à une période de temps donnée. Au début du processus, une chambre peut ne pas encore avoir été louée.

Enfin, on décrit les **données** des objets et des liens.

- Un *client* est caractérisé par son *nom*, son *adresse* et son *numéro* de téléphone.
- Une *chambre* est caractérisée par son *numéro*, un *nombre de places*, son *tarif* journalier et la présence ou non d'un *cabinet de toilettes*.
- Une location est caractérisée par une date de début, un nombre de jours et les consommations annexes (petits déjeuners, téléphone…).
- Une *réservation* est caractérisée par une *date de début*, un *nombre de jours* et le versement d'une *avance* éventuelle.

2 Modélisation par le modèle entité-association

Après l'étape de recueil d'informations, on dispose d'un ensemble de phrases simples qui expriment les besoins décrivant la réalité à modéliser : on doit alors en effectuer une représentation. Cette dernière est très importante, car elle est la seule qui donnera une vue d'ensemble des données et des liens qui les caractérisent. Le modèle obtenu à cette étape est en général nommé **Modèle Conceptuel des Données (MCD)**. En effet, on verra lors de l'étape du passage au modèle relationnel, appelé également **modèle logique**, que cette information n'apparaît plus. Les données stockées dans le SGBD seront presque inutiles si l'on ne dispose pas du modèle conceptuel qui est l'équivalent du schéma technique d'un appareil ou du plan d'un bâtiment.

Le formalisme le plus répandu pour constituer ce schéma est le **modèle entité-association** (ou *entity-relationship* en anglais). Il a été présenté à l'origine par P. Chen en 1976 aux États-Unis quasi simultanément avec le modèle de H. Tardieu en France, ce qui explique les notations légèrement différentes en Europe et aux États-Unis, en particulier au niveau de la représentation des cardinalités. Le modèle entité-association a été normalisé à l'ISO. On verra dans la section suivante que l'on peut utiliser également le formalisme **UML** pour la représentation.

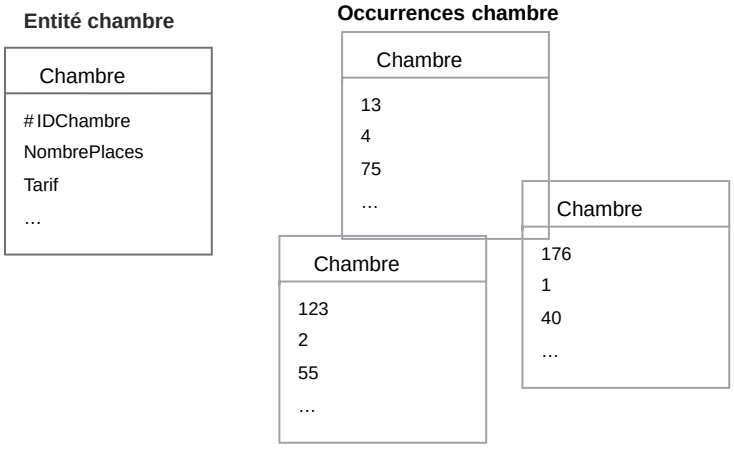
Le formalisme « entité-association », tout comme UML, utilise une représentation graphique sous forme de diagrammes. Les entités sont les objets concrets ou abstraits du monde réel évoqués plus haut. Les associations représentent le lien entre ces entités. Comme on l'a vu précédemment, on peut identifier les entités et les associations en effectuant une analyse du discours, c'est-à-dire des phrases de type « sujet-verbe-complément ». Les sujets et les compléments sont les entités, et le verbe modélise l'association.

2.1 ENTITÉS

Les **entités** sont composées de champs de données que l'on nomme **attributs**. Un attribut, ou un ensemble d'attributs, doit être choisi comme **identifiant** de l'entité, avec pour objectif d'identifier une occurrence (ou représentant) de cette entité. La notion d'identifiant a les mêmes propriétés que la clé dans une relation qui sera introduite au chapitre 3. On représente une entité par un rectangle qui contient le nom de l'entité et ses attributs. L'identifiant est souligné ou précédé d'un caractère '#' (voir figure 2.1).

Figure 2.1

Entités 'chambre', 'attributs' et 'occurrences'.

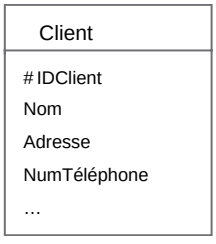


Le choix de l'identifiant n'est pas toujours trivial. Il est parfois nécessaire d'introduire artificiellement un attribut supplémentaire afin de pouvoir disposer d'un identifiant. Dans le cas de l'hôtel, il faudrait intégrer un attribut pour identifier un client (voir figure 2.2), dans la mesure où aucun des attributs issus de l'analyse ne permet d'identifier de manière unique un client. Classiquement, une identification sans ambiguïté reposera sur un numéro unique ; dans notre exemple, c'est l'attribut IDClient.

Les identifiants peuvent être composés par la juxtaposition de différents attributs. Par exemple, on peut identifier un client en juxtaposant les attributs nom+prénom+date_naissance+ville_naissance. En effet, il est peu probable que deux homonymes soient nés le même jour dans la même ville. Cependant, dans la pratique, il est recommandé, autant que faire se peut, de choisir un seul attribut comme identifiant. En effet, il sera plus difficile de vérifier qu'un identifiant composite reste valide lorsque les données évoluent. C'est pourquoi, quand cela est possible, il est indispensable de choisir un identifiant dont les contenus ne sont pas susceptibles d'évoluer au fil du temps. On préfère identifier une personne par un numéro de sécurité sociale que par un numéro de passeport qui a une durée de validité limitée.

Figure 2.2

Entité 'client'.



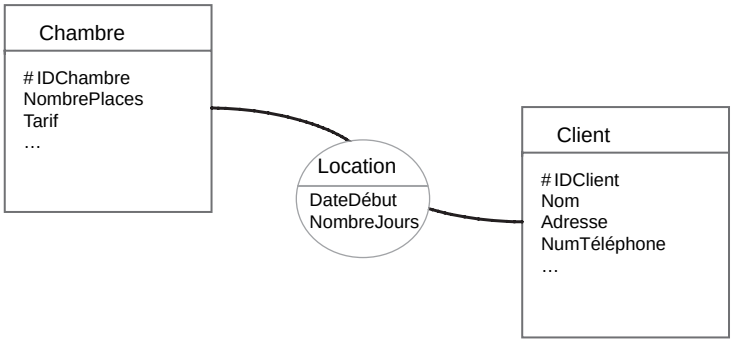
2.2 ASSOCIATIONS

Les **associations** représentent les liens qui existent entre les entités. Elles sont composées le cas échéant d'attributs, bien que cela ne soit pas indispensable. Par conséquent, il n'est pas nécessaire de disposer d'un identifiant pour une association. Lorsque les entités sont associées par deux, elles sont qualifiées de **binaires**. Cependant, il est possible d'en associer plus de deux ; les associations sont alors non plus **binaires**, mais **n-aires**. Le nombre d'entités associées s'appelle le **degré** de l'association.

On représente une association par un ovale qui contient le nom de l'association et ses attributs.

Figure 2.3

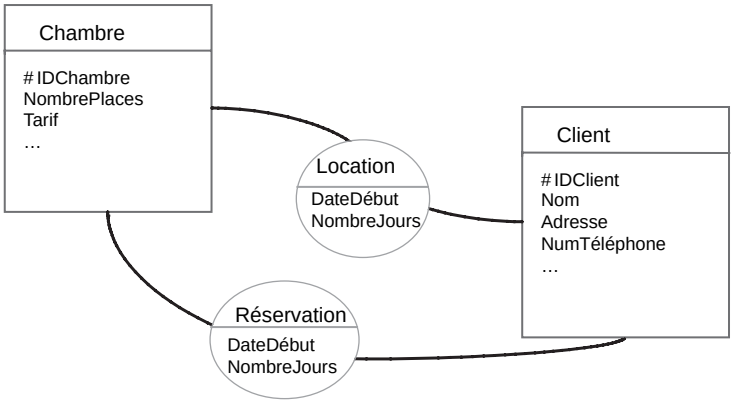
Entités 'client' et 'chambre 'reliées par l'association 'location'.



Il peut également y avoir plus d'une association entre deux entités ; c'est le cas de l'exemple de l'hôtel (voir figure 2.4). Les entités 'client' et 'chambre' sont reliées par deux associations ayant des attributs différents : 'location' et 'réservation'.

Figure 2.4

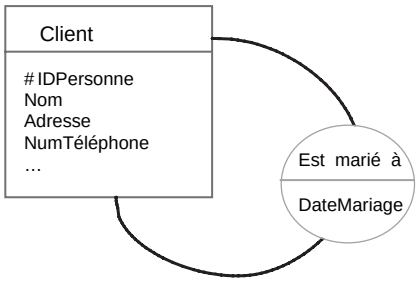
Entités 'client' et 'chambre' reliées par les associations 'location' et 'réservation'.



Enfin, il est possible de relier par une association une entité à elle-même. Si l'on prend l'exemple de la modélisation des liens de mariage entre personnes, on obtient une seule entité 'personne' qui est associée à elle-même par l'association 'est_marié_à' (voir figure 2.5). Dans ce cas, on dit que **l'association est réflexive**.

Figure 2.5

Entité 'personne' reliée par l'association 'est_marié_à'.

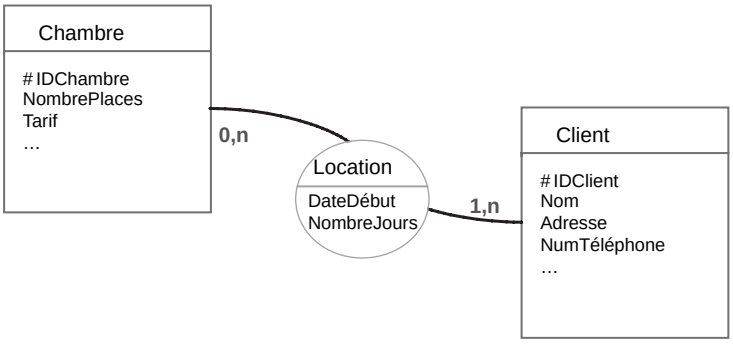


2.3 CARDINALITÉS

Les **cardinalités** décrivent les caractéristiques de l'association entre les entités. On utilise deux nombres qui représentent les valeurs minimales et maximales pour caractériser l'association. Ces nombres modélisent le nombre d'occurrences minimales et maximales des entités impliquées dans l'association. Par exemple, comme illustrée sur la figure ci-après (voir figure 2.6) , une association binaire sera caractérisée entièrement par quatre nombres. En effet, chaque entité participe de manière différente à l'association.

Figure 2.6

Cardinalités d'une association binaire.



Les cardinalités peuvent prendre les valeurs suivantes :

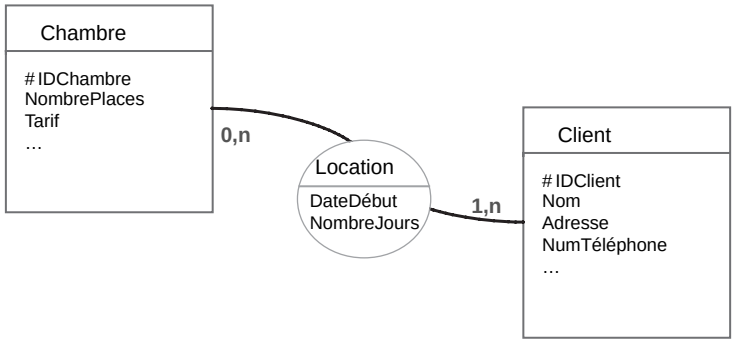
- De un à un, notée **1,1**. Une brosse à dents possède en théorie un (1) et un (1) seul propriétaire.
- De un à plusieurs, notée **1,n**. Un livre a au moins un (1) auteur ; il peut en posséder plusieurs (n). On ne considère pas les ouvrages anonymes.
- Optionnel, notée **0,1**. Une personne est célibataire (0) ou mariée (légalement…) à une (1) autre personne au plus.
- De zéro à plusieurs, notée **0,n**. Un appartement peut être libre (0) ou habité éventuellement par plusieurs habitants (n).

Dans l'exemple de l'hôtel précédent, l'analyse préalable permet de déduire les propriétés suivantes (voir figure 2.7) :

- Un client loue au minimum une (1) chambre ; il peut en louer plusieurs (n). La cardinalité est donc de 1,n.
- Une chambre peut être louée plusieurs fois (n) et elle peut ne pas être occupée (0). La cardinalité est donc de 0,n.

Figure 2.7

Entités 'client', 'chambre' et association 'location' avec cardinalité.



Remarque

Les cardinalités se notent différemment dans le modèle de Chen employé aux États-Unis et dans celui utilisé en Europe. Dans le modèle européen, on dispose les cardinalités du côté de l'entité concernée. La cardinalité '0,n' déduite de « une chambre peut être louée plusieurs fois et elle peut ne pas être occupée » se trouve du côté de l'entité 'chambre'. Cette notation présente l'avantage d'être plus cohérente lors de l'utilisation d'associations « n-aires ». Il n'y a pas de changement dans l'emplacement des cardinalités des entités associées.

2.4 Cas d'étude « casse »

On a présenté dans le chapitre d'introduction la base de données exemple « casse », qui décrit l'activité de vente de voitures d'occasion. Voici sa modélisation sous forme de modèle entité-association. On rappelle la description de ce système.

Deux entités du monde réel sont identifiées : les personnes et les voitures. Une voiture est caractérisée par sa marque, son type, sa couleur. Une personne est caractérisée par son nom, son âge, sa ville, son sexe. L'action modélisée est la vente, caractérisée par le prix de la vente et sa date. Une personne peut acheter plusieurs voitures ou aucune. Une personne peut acheter aucune ou plusieurs voitures. Une voiture peut être vendue ou non.

Recherche des identifiants

Les deux entités 'voiture' et 'clients' ainsi que leurs attributs respectifs sont bien définis ; il manque pour chacune de ces entités un identifiant capable de différencier les occurrences (ou représentants) des entités voitures et des clients. Sur cet exemple simple, il est facile de vérifier qu'aucun attribut ni ensemble d'attributs ne permet de définir sans ambiguïté un identifiant. Le processus de recherche d'un identifiant à partir des relations de dépendances entre les attributs, est détaillé au chapitre 3.

Recherche des cardinalités

Les cardinalités sont déduites des deux dernières phrases :

- **0,n.** Pour « une personne peut acheter plusieurs (n) voitures ou aucune (0) ».
- **0,1.** Pour « une voiture peut être vendue une seule fois (1) ou jamais (0) ».

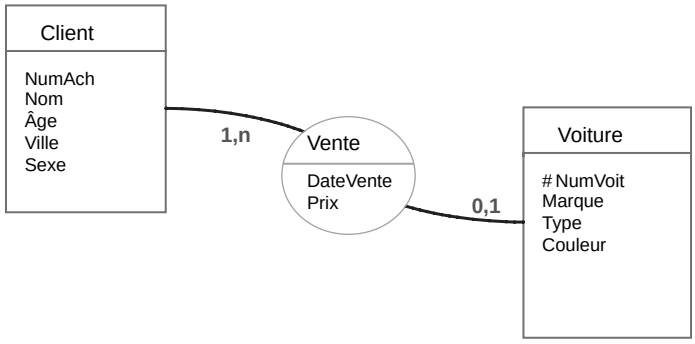


Figure 2.8

Entités 'voiture' et 'client' liées par l'association 'vente' avec cardinalités.

3 Remise en cause et évolution du modèle

Cette section énonce quelques conseils pour améliorer et affiner le modèle conceptuel obtenu à l'étape précédente.

3.1 Qualité des attributs

Dans cette sous-section, on aborde les qualités générales que doivent posséder les attributs des entités et des associations. Des règles simples permettent de guider le choix des attributs.

Une première règle est de ne stocker que les **attributs** strictement **nécessaires** : la tendance des utilisateurs est en général de prévoir le plus d'attributs possible juste au cas où. Le choix s'effectue en tenant compte des fonctionnalités attendues du système d'information par élimination systématique des données non utiles.

Une deuxième règle consiste à ne pas retenir les **attributs** qui **peuvent être déduits** d'autres attributs. La déduction peut se faire soit par le calcul, soit par un lien sémantique entre plusieurs attributs :

- Le chiffre d'affaires peut être calculé à partir du prix de vente et de la quantité.
- Une référence peut être construite à partir du nom du produit et du fournisseur.

Le but est de réduire la place occupée ainsi que les risques d'incohérence.

Enfin, une dernière règle concerne le nom des **attributs** qui doivent être « **parlants** ». Il ne faut pas hésiter à utiliser des noms de grande taille si cela facilite la compréhension du rôle de l'attribut dans l'entité. Par exemple, il est préférable d'utiliser un attribut 'Numéro_Série' plutôt qu'un attribut 'ns'. Dans la mesure du possible, on essaie de donner des noms d'attributs spécifiques pour chacune des entités. Ces deux dernières précautions faciliteront la compréhension générale du modèle et son utilisation future dans le SGBD.

3.2 Réorganisation des entités

Dans cette sous-section, on remet en cause le découpage en entités produits par une première approche. Le processus itératif de cette remise en cause peut conduire à un **redécoupage** des entités ou à leur **fusion**.

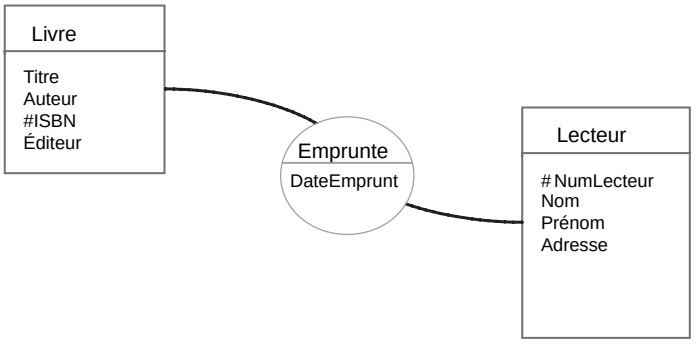
On considère l'exemple de la bibliothèque de prêt, que l'on a déjà utilisé précédemment. La phrase caractéristique décrivant l'activité de la bibliothèque est la suivante : « Un lecteur

emprunte un livre. » On en déduit les deux entités 'lecteur' et 'livre' liées par l'association 'emprunte' (voir figure 2.9).

- Un lecteur est caractérisé classiquement par son nom, son prénom et son adresse. Comme il n'y a pas d'attribut possédant les caractéristiques d'un identifiant pour l'entité 'lecteur', on ajoute un attribut identifiant. Cet attribut supplémentaire est le numéro de lecteur. À noter que dans le monde réel, ce numéro pourrait correspondre par exemple à un numéro de carte de lecteur.
- Un livre est caractérisé par son titre, son auteur, son numéro ISBN, son éditeur. Le numéro ISBN est ici un identifiant pour l'entité puisqu'il est unique.
- L'association 'emprunte' a pour attribut la date d'emprunt.

Figure 2.9

Entités 'livre' et 'lecteur' liées par l'association 'emprunte' (sans cardinalités).



Cet exemple simple permet de se poser quelques questions qui vont conduire à une réorganisation du modèle.

Que se passe-t-il si l'on possède plusieurs exemplaires du même ouvrage ?

Le numéro ISBN n'est plus identifiant puisqu'il est le même pour chacun des exemplaires. Une solution consiste à ajouter un numéro supplémentaire unique pour chaque livre, qui correspond à la notion de cote dans une bibliothèque. Ainsi, même si l'on a dix exemplaires d'un ouvrage, il est possible de les différencier. L'inconvénient de cette solution est que l'on répète les informations communes aux différents ouvrages (titre, auteur…) à chaque exemplaire. L'un des risques est de répéter incorrectement ces informations et d'aboutir ainsi à des incohérences.

La solution correcte dans ce cas est de séparer l'entité 'livre' en deux entités 'livre' et 'ouvrage' (voir figure 2.10). L'activité de la bibliothèque est alors décrite par deux phrases :

- Un lecteur emprunte un exemplaire.
- Un livre représente un exemplaire d'un ouvrage.

Un livre est caractérisé par un numéro (cote) identifiant et un ISBN. Un ouvrage est caractérisé par un ISBN, qui est bien dans ce cas un identifiant, un titre, un auteur et un éditeur. Dans cet exemple, l'objet du modèle conceptuel 'livre', utilisé comme entité, est décomposé en deux autres entités dont une est abstraite : l'ouvrage. L'ouvrage est un regroupement d'attributs qui n'a pas d'existence « tangible » dans le monde réel.

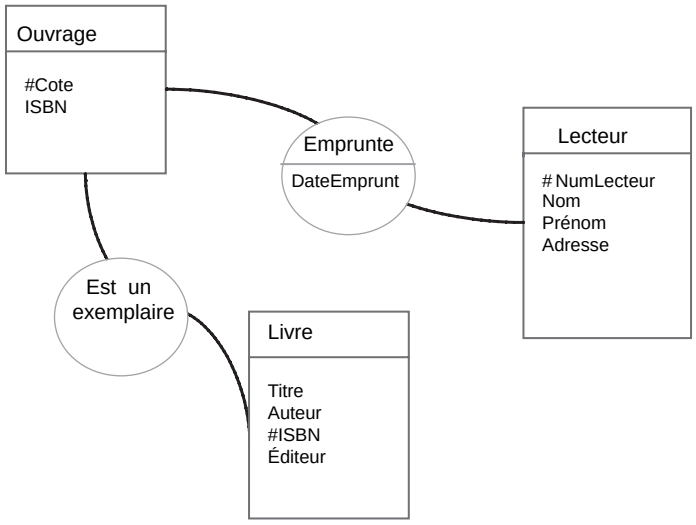


Figure 2.10

Entités 'livre', 'ouvrage' et 'lecteur' liées par les associations 'emprunte' et 'est un exemplaire' (sans cardinalités).

Que se passe-t-il si l'ouvrage possède plusieurs auteurs ?

Une solution simpliste est de prévoir un champ par auteur supplémentaire, c'est-à-dire d'ajouter des champs 'auteur2', 'auteur3', 'auteur4', etc. Cette solution pose de nombreux problèmes :

- Si seulement dix livres sur un million possèdent plusieurs auteurs, on réserve la place pour les champs auteurs supplémentaires qui sera inutilisée.
- Si un livre possède un nombre d'auteurs supérieur au nombre de champs prévus, on ne résout pas le problème.
- Si l'on considère qu'un auteur peut avoir écrit plusieurs ouvrages, on répète dans ce cas les informations le concernant pour chacun de ses ouvrages. Cela constitue un cas typique de redondance qui risque de provoquer des incohérences.

La solution correcte dans ce cas est de créer une entité supplémentaire pour ces attributs qui sont sémantiquement de même type. On créera ici une entité auteur qui contiendra un numéro d'auteur (identifiant), son nom et son prénom. Cette entité est reliée à l'entité ouvrage par l'association 'a_écrit' (voir figure 2.11).

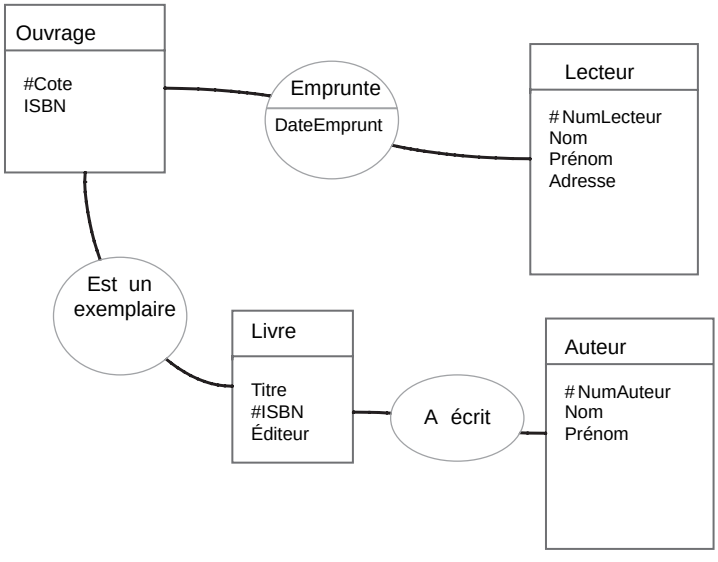


Figure 2.11

Entités 'livre', 'ouvrage', 'auteur' et 'lecteur' liées par les associations 'emprunte', 'a_écrit' et 'est un exemplaire' (sans cardinalités).

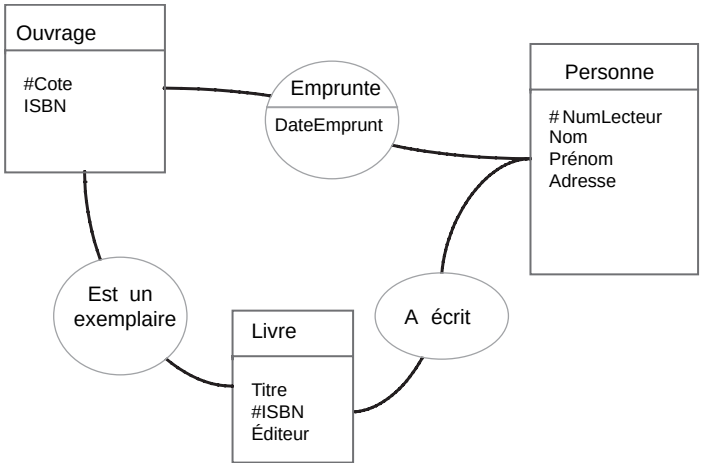
Que se passe-t-il si un auteur emprunte un livre ?

Un auteur peut également emprunter un livre et revêtir par conséquent le rôle de l'emprunteur. Dans ce cas, on répète les informations le concernant dans l'entité 'lecteur'. De même que précédemment, cela provoque de la redondance et peut générer des incohérences.

Comme les entités 'lecteur' et 'auteur' ont la même structure (c'est-à-dire des mêmes attributs), la solution consiste à les fusionner en une entité unique 'personne'. Cette opération conduit à une association entre les entités 'personne' et 'livre' (emprunte) et les entités 'personne' et 'ouvrage' (a_écrit) (voir figure 2.12).

Figure 2.12

Entités 'livre', 'ouvrage', 'personne' liées par les associations 'emprunte', 'a_écrit' et 'est_un_exemplaire' (sans cardinalités).



On a identifié dans cette sous-section plusieurs cas qui nécessitent de réorganiser les entités obtenues lors d'une première analyse :

- Si l'on repère à l'intérieur d'une entité des attributs qui représentent un ensemble logique (mis en valeur dans l'exemple par un défaut d'identifiant pour un livre), on sépare ces attributs dans une entité supplémentaire.
- Si des attributs dans une même entité possèdent une sémantique identique (auteurs, numéros de téléphone multiples), on crée une entité supplémentaire pour séparer ces attributs. Cette entité sera associée à celle dont proviennent les attributs à l'origine.
- Si des entités ont la même structure (et représentent le même type d'objet), on les fusionne et l'on conserve les associations qui existaient avant la fusion.

On retrouvera ce processus de remise en question du modèle à un autre niveau, lors de l'étape de normalisation qui est abordée au chapitre 3.

3.3 Élimination et fusion d'associations

Dans cette sous-section, on s'intéresse à la réorganisation d'associations qui n'ont pas de raison d'être, par une fusion des associations ou par leur intégration dans les entités.

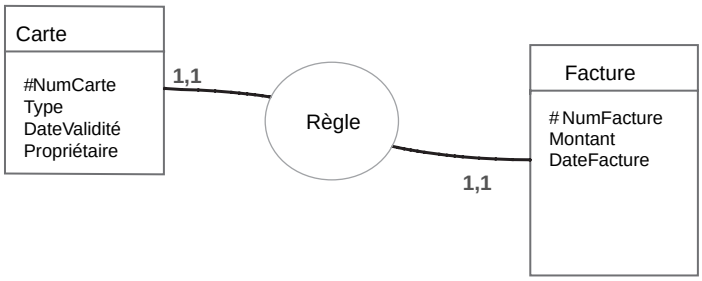
Élimination d'associations

On considère le cas d'un acte d'achat effectué sur Internet avec une carte bancaire. Une facture est réglée par une carte. On peut distinguer les deux entités 'facture' et 'carte' . Une facture est identifiée par un numéro de facture, et est constituée d'un montant et d'une date. Une carte bancaire est identifiée par son numéro, son type (Visa, MasterCard), sa date de validité et son propriétaire. Les cardinalités entre les entités 'carte' et 'facture' sont de type 1-1 (une facture est payée par une et une seule carte) et 1-n (une

carte peut servir à régler plusieurs factures). On peut utiliser une Ecarte qui permet d'améliorer la sécurité de ces transactions. Une Ecarte est une carte « virtuelle » associée à une véritable carte bancaire, valable pour une seule transaction. Les cardinalités deviennent alors de type 1-1 des deux côtés : une Ecarte ne permet de régler qu'une et une seule facture (voir figure 2.13).

Figure 2.13

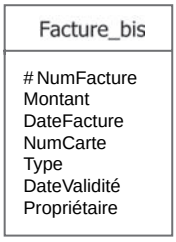
Entités 'carte' et 'facture' liées par l'association 'règle' avec ses cardinalités.



Dans ce cas, l'association 'règle' n'a plus lieu d'être puisqu'il s'agit d'une pure bijection : à une facture correspond une Ecarte et une seule, et à une Ecarte correspond un produit et un seul. On peut fusionner les deux entités 'carte' et 'facture' et éliminer l'association (voir figure 2.14).

Figure 2.14

Entité 'facture_bis' fusionnée.

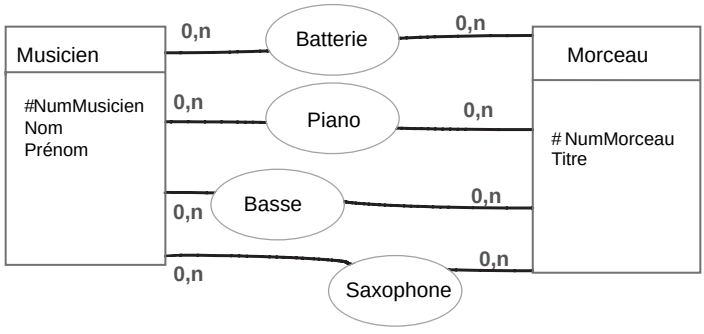


Fusion d'associations

Suivant le même principe, il est possible de fusionner plusieurs associations ayant le même rôle sémantique. Si l'on considère la description de l'activité suivante, liée à l'exécution de morceaux de jazz en quartet. Pour un morceau donné, le premier musicien joue la partie de basse, le deuxième celle de batterie, le troisième celle de piano et le quatrième celle de saxophone (voir figure 2.15).

Figure 2.15

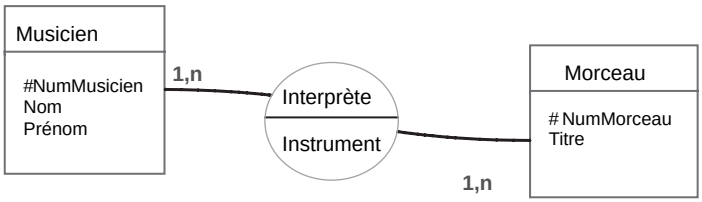
Entités 'musicien' et 'morceau' liées par les associations 'basse', 'batterie', 'piano' et 'saxophone'.



Comme il s'agit de la même activité (un musicien interprète un morceau), on peut remplacer toutes les associations par une association 'interprète'. L'association 'interprète' contiendrait l'attribut 'instrument' pour ne pas perdre l'information de l'instrument associé (voir figure 2.16).

Figure 2.16

Entités 'musicien' et 'morceau' liées par l'association 'interprète'.



Remarque

Il en serait différemment pour des associations entre ces deux entités qui exprimeraient un autre sens. Si l'on considère l'association 'compose' ou 'arrange' entre les entités 'musicien' et 'morceau', il ne s'agit pas de la même activité que l'association 'interprète' : on ne fusionnera pas les associations dans ce cas.

Dans cette section, on a essayé de porter un regard critique sur le modèle conceptuel « brut » issu d'une première phase d'analyse que l'on a obtenue en utilisant les techniques énoncées dans les sections précédentes. Cette étape fait partie intégrante du processus itératif de constitution du modèle par raffinements successifs. Le processus se poursuivra à un autre niveau par la normalisation des relations déduites de ce modèle. Cette partie sera traitée au chapitre 3.

4 Représentation avec UML

Le langage UML (*Unified Modeling Language*) est un standard élaboré à l'initiative de l'OMG (*Object Management Group*). Il est donc spécifiquement destiné à la modélisation objet. Comme son nom l'indique, UML est le résultat d'un processus de fusion (unification) de différentes méthodes existantes dans le domaine de la conception objet : Booch, OMT, OOSE. Ces concepts reposent principalement sur les travaux de Grady Booch, James Rumbaugh et Ivar Jacobson. C'est un langage indépendant de la plate-forme utilisée ainsi que des autres langages de plus bas niveau employés en programmation objet, comme Java et C++. Il utilise une notation graphique et a connu très rapidement le succès dans le monde objet par la souplesse qu'il apporte en gestion de projets de développement.

4.1 UML et bases de données

Quel est le rapport avec les bases de données ? UML a été adapté afin d'être utilisé pour la modélisation en bases de données : on recourt à cet effet à un profil spécifique (*UML profile for data modeling*). Même s'il est prévu initialement uniquement pour la modélisation objet, UML permet aussi de représenter un modèle conceptuel utilisable pour créer des bases de données relationnelles et relationnelles-objet. Les approches objet et relationnelles-objet en bases de données ont été introduites dans le premier chapitre et dépassent largement le cadre de cet ouvrage.

En outre, une motivation supplémentaire pour utiliser UML est de pouvoir disposer d'outils logiciels graphiques du marché pour la description et l'automatisation du processus de passage au SGBD, allant jusqu'à la génération de code SQL. UML offre des garanties de pérennité du fait de sa normalisation. Même si UML n'apporte pas d'évolutions majeures quant aux méthodes existantes de modélisation en bases de données, il est de plus en plus largement répandu et permet d'utiliser le même formalisme et les mêmes outils que les concepteurs d'applications logicielles.

Remarque

UML est un formalisme de représentation. Il n'est associé à aucune méthode particulière comme peut l'être la méthode Merise qui associe une méthodologie et un langage de description.

4.2 Représentation du modèle conceptuel avec les diagrammes de classes UML

UML a été développé à l'origine pour la modélisation des concepts objet. La tendance actuelle consiste à utiliser une partie restreinte de ce puissant langage de modélisation, afin de représenter le modèle conceptuel des données. Dans sa version complète, UML propose neuf types de diagrammes répartis en deux catégories : les diagrammes de **structure** qui décrivent la partie statique du modèle ; les diagrammes de **comportement** qui décrivent la partie dynamique, c'est-à-dire les traitements. On utilise les **diagrammes de classes** qui font partie des diagrammes de structure pour représenter les éléments du modèle conceptuel.

Classes

Dans ce type de représentation les entités sont interprétées comme des **objets** et sont représentées par des **classes**. La description d'une classe en UML comprend trois parties :

- le nom de la classe ;
- la description des attributs ;
- les méthodes associées à l'objet.

Cette section n'utilise pas la notion de méthode spécifique à l'approche objet. La description des attributs peut être réalisée de manière plus précise. UML offre la possibilité de décrire à ce niveau le domaine de l'attribut :

jour : (lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche)

Une classe est représentée graphiquement par un rectangle séparé en trois zones correspondant aux trois parties précédentes (voir figure 2.17).

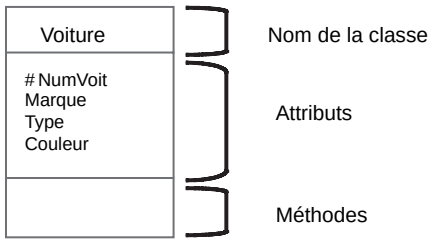


Figure 2.17

Entité 'voiture' représentée par une classe UML.

Associations et multiplicité

Les classes sont reliées par des **associations** identifiées par leur nom. Si l'association possède elle-même des attributs, ils sont intégrés dans une classe d'association. L'expression des cardinalités est quasiment la même en UML que pour le modèle entité-association ; elles sont appelées **multiplicité**. UML permet de définir plus précisément les cardinalités en utilisant les valeurs suivantes.

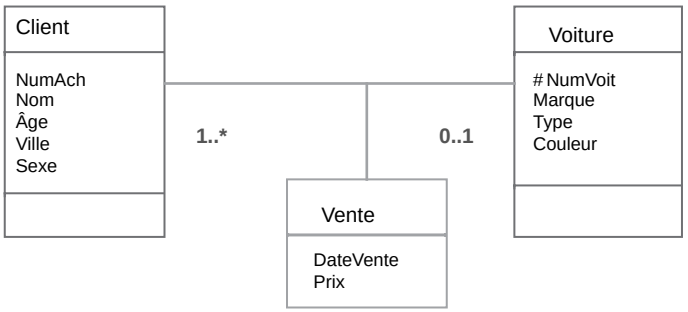
- '1'. Obligatoire, un et un seul (peut s'écrire également 1..1).
- '*'. Non obligatoire (peut s'écrire également 0..n).

- '0..1'. Non obligatoire restreint à 1.
- '1..*'. Obligatoire.
- '2..4'. Obligatoire restreint de 2 à 4.

Les associations sont matérialisées par un trait entre les classes représentant les entités. Le nom de l'association est inscrit au-dessus du trait. Dans le cas où l'association possède des attributs, on utilise une classe d'association sans nom qui contiendra les attributs. La classe d'association est reliée par un trait pointillé (voir figure 2.18).

Figure 2.18

Entités 'voiture' et 'client' liées par l'association 'vente' représentées par une classe UML.



Remarque

Attention, la position des cardinalités dans un diagramme de type UML est inversée par rapport à celle utilisée dans un diagramme de type modèle entité-association présentée dans ce chapitre. En effet, UML utilise la notation employée aux États-Unis, mal adaptée à la représentation d'association de degré supérieur aux associations **binaires**.

On peut décrire plus finement l'association en spécifiant un **rôle** de l'association pour chacune des classes associées. C'est-à-dire que l'on distingue, du point de vue de chacune des classes impliquées dans l'association, la manière dont elle est associée à une autre classe par un terme spécifique : le rôle. Comme les cardinalités sont différentes pour chacune des classes impliquées dans une association, on dispose alors d'une description complémentaire. La notion de rôle est surtout utile lorsqu'une classe est associée à elle-même.

Par exemple, si l'on considère les classes 'voiture' et 'client' (voir figure 2.19), du point de vue de la classe 'voiture', la classe 'client' a le rôle d'acheteur. Du point de vue de la classe 'client', la classe 'voiture' a le rôle d'une dépense. Ainsi, on note le nom du rôle au même emplacement que celui de l'association (au-dessus du trait), mais du côté de la classe concernée.

Figure 2.19

Classes 'voiture' et 'client' liées par l'association 'vente' avec des rôles.

4.3 Du modèle entité-association à UML

La notation UML est utilisée dans cette section pour représenter les entités et les associations issues de la démarche d'analyse « primitive » que l'on a décrite dans la première sec-

tion. La correspondance qui est faite avec celle fondée sur le modèle entité-association est la suivante :

- Une entité est représentée par une classe.
- Une association est représentée par une association.
- Une cardinalité est représentée par une multiplicité.

À ce niveau d'utilisation, les avantages qu'apporte la notation UML n'apparaissent pas clairement. L'intérêt réside dans la possibilité d'utiliser des outils logiciels. Ces derniers, à partir du schéma du modèle conceptuel exprimé en UML, automatisent, ou plutôt assistent, le processus de passage au modèle relationnel qui sera abordé au chapitre 3.

L'idée à terme est de générer automatiquement le langage SQL nécessaire, comme cela se fait en génie logiciel où l'on génère le code du langage utilisé. En revanche, l'utilisation d'UML prendra tout son sens pour une modélisation plus complexe utilisant par exemple le concept objet d'héritage qui dépasse le cadre de cet ouvrage.

Résumé

Ce chapitre aborde l'approche de la modélisation, qui consiste d'abord à extraire les informations du monde réel puis à en proposer une représentation formelle appelée modèle conceptuel. Il n'existe pas réellement de démarche « scientifique » pour réaliser cette étape. Elle est plutôt constituée d'un ensemble de techniques diverses qui permettent d'appréhender la complexité du monde réel à modéliser et de la simplifier.

L'analyse du monde réel a pour but de pouvoir identifier les données qui interviennent dans le domaine considéré et de les regrouper dans des objets. L'étape suivante consiste à caractériser les liens qui les unissent. À cette fin, on décrit le système à modéliser par des phrases simples de type « sujet-verbe-complément », que l'on peut classer en deux grandes catégories :

- celles qui décrivent les objets du monde réel et les liens qui les unissent : le sujet et le complément sont représentés par des **entités** (classes) et le verbe est représenté par une **association ;**
- celles qui décrivent la manière dont sont reliés ces objets : on en déduira les cardinalités (ou multiplicités dans le cas d'utilisation d'UML) des associations.

Les entités sont constituées de champs de données que l'on nomme « attributs ». Pour identifier de manière unique les représentants d'une entité, appelés également « instance », un attribut (ou un ensemble d'attributs) est choisi : on l'appelle « identifiant ». Une fois les éléments à modéliser identifiés, leur représentation normalisée recourt à différents langages. Les deux types de formalismes les plus courants employés sont les suivants :

- **Le modèle entité-association.** Il a fait ses preuves ; sa notation est très intuitive et il est encore couramment utilisé.
- UML (*Unified Modeling Language*). Il représente (probablement) l'avenir, car il est soutenu par une communauté très importante qui dépasse celle des bases de données. Il offre l'avantage d'être bien adapté à la modélisation de données complexes qui nécessitent une approche objet.

Ni le modèle entité-association, ni UML ne sont des méthodes d'analyse ; il s'agit dans les deux cas de simples formalismes de représentation.