CHAP I

Introduction aux bases de données

Au cours des dernières années, les bases de données ont connu un développement considérable, au point qu'elles jouent désormais un rôle dans chacune de nos opérations quotidiennes – du simple achat effectué avec sa carte bancaire jusqu'à nos déclarations de revenus.

L'objectif de ce chapitre est de définir la notion de base de données ainsi que les principaux concepts qui s'y rattachent. La méthodologie qui permet de les concevoir, les applications informatiques associées à leur mise en œuvre (SGBD) et les différents métiers des bases de données y sont présentés.

1 Qu'est-ce qu'une base de données ?

Le nombre d'informations disponibles et les moyens de les diffuser sont en constante progression. La croissance du *World Wide Web* a encore accru ce développement, en fournissant l'accès à des bases de données très diverses avec une interface commune. Celles-ci se situent au cœur de l'activité des entreprises, des administrations, de la recherche et de bon nombre d'activités humaines désormais liées à l'informatique.

Dans le domaine purement informatique, elles interviennent dorénavant à tous les niveaux. Les développeurs d'applications s'appuient sur des bases de données externes pour gérer leurs données alors qu'auparavant elles étaient intégrées dans le programme. Citons un autre exemple : la gestion des fichiers dans les nouveaux systèmes d'exploitation (par exemple, Panther de chez Apple ou le futur Vista de Microsoft) évolue vers de véritables bases de données mises à jour en permanence. Elles permettent de retrouver les fichiers instantanément, par leur nom mais aussi par leur contenu, à la manière d'un moteur de recherche.

Les **bases de données** reposent sur des théories solides et sont à l'origine d'une des plus importantes disciplines de l'informatique : l'**ingénierie des systèmes d'information**. Cette section présente une idée intuitive de ce qu'est une base de données, de son utilisation puis des éléments de qualité qui lui sont associés.

1.1 NOTION DE BASE DE DONNÉES

Tout le monde a une idée naturelle de ce que peut être une base de données : elle peut revêtir la forme d'une liste de CD contenant le nom des artistes et les titres des morceaux ou encore celle de fiches de recettes de cuisine.

On remarque qu'une caractéristique des données contenues dans une base de données est qu'elles doivent posséder un lien entre elles. En effet, des données choisies au hasard ne constituent certainement pas une base de données. Celle-ci est donc une représentation partielle et (très) simplifiée du monde réel, que l'on a obtenu par un processus de **modélisation**. En résumé, on définit une base de données comme l'ensemble des données stockées. Pour les manipuler, on utilise généralement un logiciel spécialisé appelé SGBD (*Système de Gestion de Bases de Données*). Il y a parfois confusion, par abus de langage, entre base de données et SGBD. On appelle aussi « système d'information » l'ensemble composé par les bases de données, le SGBD utilisé et les programmes associés. Plus formellement, on appelle ***Base de Données*** (BD) un ensemble de fichiers – informatiques ou non – structurés et organisés afin de stocker et de gérer de l'information.

1.2 UTILISATION D'UNE BASE DE DONNÉES

La création d'une base de données recèle un but précis : elle doit permettre de retrouver de l'information par son contenu en se fondant sur des **critères de recherche**. On désire, par exemple, retrouver toutes les recettes qui nécessitent des œufs ou tous les CD qui contiennent un morceau donné.

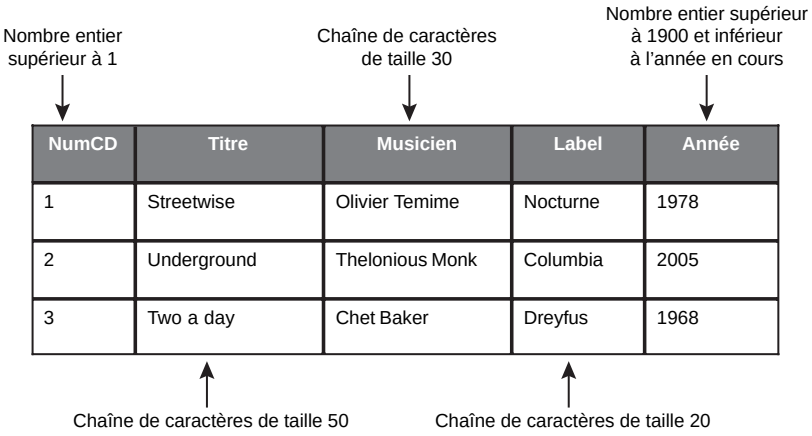
La grande différence avec un programme écrit dans un langage de programmation est qu'une base de données doit pouvoir répondre à des questions pour lesquelles elle n'a pas forcément été prévue à la conception.

Une autre différence est que les données sont susceptibles d'être utilisées par des applications différentes. Dans un programme classique, la structuration des données est décrite

directement dans le code, ce qui rend leur utilisation difficile par d'autres programmes, en particulier lorsque l'on modifie cette structure. Ce que l'on recherche en utilisant une base de données est d'assurer l'indépendance entre le traitement et les données. C'est pourquoi, il est nécessaire que l'application obtienne des informations sur la structure des données (nom, type, taille, etc.). Pour ce faire, on associe à la base de données une description que l'on appelle « **métadonnée** » ou « **catalogue** ». Cette dernière décrit la structure interne de la base de données qui est spécifique au SGBD employé (voir figure 1.1). En plus de la structure et du type des données, on stocke également à cet endroit les informations concernant les règles de cohérence des données abordées à la section suivante.

Figure 1.1

Base de données de CD et métadonnée.



L'idée générale est que l'utilisateur ou l'application utilisatrice des données ne doit pas être dépendante de leur représentation interne, ce qui constitue une abstraction des données. C'est la raison pour laquelle on utilise une description des données sous la forme d'un modèle pour permettre la restitution la plus efficace possible de l'information.

1.3 Qualité d'une base de données

Comme on l'a évoqué précédemment, l'un des objectifs de création d'une base de données est de pouvoir retrouver les données par leur contenu. Dans cette optique, il faut s'assurer que les données contenues dans la base sont de « bonne qualité ».

Comment définir la qualité des données ? De nombreux critères peuvent être pris en compte ; on peut citer parmi les principaux :

- la cohérence des données contenues dans la base ;
- l'absence de redondance.

La **cohérence des données** est fondamentale ; elle nécessite une réflexion préalable sur la normalisation du contenu des champs. On suppose qu'un champ contient la qualité d'une personne (par exemple, Monsieur, Madame, Mademoiselle). Si l'on trouve dans ce champ 'Mr' à la place de 'Monsieur', il est clair que les recherches sur ce champ par le contenu 'Monsieur' risquent d'être erronées. Dans ce cas, les informations seraient moins nombreuses que celles obtenues avec le contenu correct. On qualifie cet état de fait de « **silence** », qui signifie que certains résultats pertinents sont ignorés lors d'une interrogation.

Dans un autre cas, si l'on saisit 'Mme' pour 'Madame' et 'Melle' pour 'Mademoiselle', et qu'il y ait eu par erreur plusieurs saisies de 'Mme' alors qu'il s'agissait d'une demoiselle, la recherche par le contenu 'Mme' donne cette fois plus de résultats qu'il n'y a réellement de dames. On qualifie cet état de fait de « **bruit** », qui signifie que certains résultats non

pertinents sont retournés lors d'une interrogation. La **redondance** est parfois plus délicate à identifier. Si l'on considère le cas très simple d'un carnet d'adresses qui contiendrait en même temps le code postal et le nom de la ville, elle est ici évidente.

Tableau 1.1

Exemple de redondance de l'information.

Nom	Téléphone	Ville	Code postal
Jaco	0668541087	Bordeaux	33000
Stanley	0654789254	Nancy	54000
Marcus	0658741263	Bordo	33000
Charles	0639517720	Nancy	54000
Steve	0659874120	Boredeaux	33000

On remarque que l'on stocke plusieurs fois la même association d'information (par exemple, Nancy et 54000), ce qui consomme de la place inutilement et peut devenir significatif lorsque la base atteint quelques millions d'enregistrements.

De plus il existe des incohérences dans la saisie du nom de la ville 'Bordeaux'. La recherche par le nom 'Bordeaux' ne donnera pas le même résultat que la recherche par le code '33000'.

On verra plus loin que l'approche relationnelle procure des outils capables de détecter et d'améliorer considérablement ce genre de problèmes de qualité des bases de données.

2 Évolution des bases de données et de leur utilisation

Le développement des bases de données s'étend sur une période d'une quarantaine d'années. Cette section présente tout d'abord le contexte dans lequel elles se sont développées. On aborde ensuite les modèles utilisés successivement pour les représenter. Enfin, la dernière partie présente une vue prospective des changements dans l'utilisation des bases de données.

2.1 Contexte

À partir des années 1960, les ordinateurs évoluent rapidement. Ils sont de plus en plus performants mais aussi de plus en plus répandus du fait de leur coût plus raisonnable. Leur utilisation change également ; on passe de la notion de calculateurs purs à des machines capables aussi de traiter de l'information. On parvient à un niveau d'abstraction supplémentaire par rapport aux machines et on obtient en conséquence une indépendance par rapport à l'architecture et surtout par rapport aux constructeurs.

La décennie des années 1970 est une période « faste » pour la recherche et l'innovation en informatique dont les résultats sont encore utilisés aujourd'hui. On peut utiliser des langages de programmation de haut niveau, afin de guider, voire de façonner, la démarche du programmeur (Pascal), et l'on envisage des systèmes d'exploitation indépendants de la machine employée (Unix). C'est également à cette époque que l'on pose les fondements des techniques qui sont utilisées dans les réseaux (TCP/IP), en particulier pour Internet. C'est dans ce contexte favorable que E. F. Codd définit et développe l'approche relationnelle en base de données.

L'objectif principal est d'éloigner l'utilisateur des détails d'implémentation et de faciliter ainsi l'usage de l'informatique. Un autre but est de rendre « **génériques** » et réutilisables les développements informatiques, parfois devenus caducs en raison d'un changement de machine. Dans le domaine des bases de données, le développement de l'**architecture à trois niveaux** constitue une première étape importante. Les fonctionnalités des systèmes de bases de données sont séparées en trois niveaux : **niveau physique**, **niveau logique** et **niveau externe**.

2.2 MODÈLES

Les **modèles de données** correspondent à la manière de structurer l'information dans une base de données. Ils reposent sur les principes et les théories issus du domaine de la recherche en informatique et permettent de traduire la réalité de l'information vers une représentation utilisable en informatique.

Modèle hiérarchique et modèle réseau

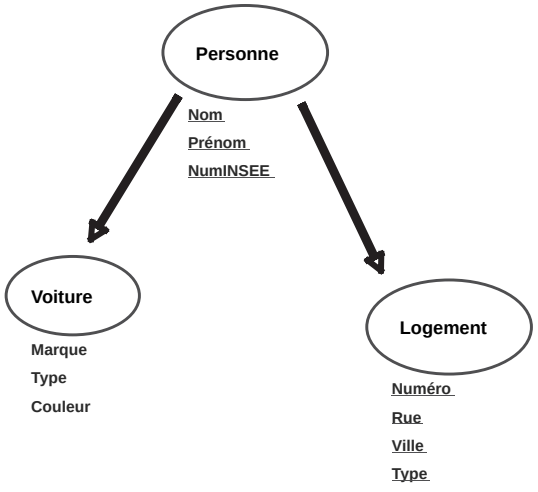
Le traitement de l'information à cette époque est encore très lié à l'organisation des fichiers sur une machine. Les modèles conceptuels de données sont eux aussi très proches du système de fichiers puisque l'on manipule des graphes ou des arbres. Les nœuds de ces structures constituent les informations et les liens entre ces données les arêtes. À ce moment, on n'est pas encore capable de séparer complètement le **niveau logique** du **niveau physique** d'un système de bases de données (voir figure 1.2).

Le modèle « hiérarchique » propose une classification arborescente des données à la manière d'une classification scientifique. Dans ce type de modèle, chaque enregistrement n'a qu'un seul possesseur ; par exemple, une commande n'a qu'un seul client. Cependant, notamment à cause de ce type de limitations, ce modèle ne peut pas traduire toutes les réalités de l'information dans les organisations.

Le modèle « réseau » est une extension du modèle précédent : il permet des liaisons transversales, utilise une structure de graphe et lève de nombreuses limitations du modèle hiérarchique. Dans les deux cas, les enregistrements sont reliés par des pointeurs : on stocke l'adresse de l'enregistrement auquel il est lié. Des SGBD de type hiérarchique ou réseau sont encore employés pour des raisons d'efficacité lorsque la structure des données s'y prête. On utilise à cet effet des SGBD de conception ancienne, comme IMS (Bull) pour le modèle réseau ou IDMS (Computer Associate).

Figure 1.2

Modèle hiérarchique.



Modèle relationnel

En 1970, E. F. Codd propose un nouveau modèle « relationnel » dans un article resté célèbre : « A Relational Model of Data for Large Shared Data Banks », *CACM 13*, n° 6, June 1970. Il cherche à créer un langage d'interrogation des bases de données plus proche du langage naturel. Dans cette optique, il fonde sa recherche sur des concepts mathématiques rigoureux, tels que la théorie des ensembles et la logique du premier ordre. Le modèle relationnel permet de modéliser les informations contenues dans les bases de données en utilisant des **relations**, c'est-à-dire des ensembles d'**attributs** (voir figure 1.3).

De l'idée de départ à la réalisation d'un produit utilisable, le laps de temps est souvent de l'ordre d'une décennie. La mise en œuvre des idées de Codd se fait chez IBM dans le cadre du projet de recherche System-R. Le premier produit commercial sera non pas le fait d'IBM, mais celui d'Honeywell en 1976. Il sera suivi d'un produit réellement abouti de chez Relationnel Software en 1980 : Oracle, qui a connu le succès que l'on sait. De son côté IBM en tirera un produit qui deviendra DB2.

Toujours dans le cadre du projet de recherche System-R, E. F. Codd met au point, en même temps que le modèle relationnel, un langage d'interrogation des données, SEQUEL, qui deviendra ensuite **SQL** (***Structured Query Language***). La normalisation du langage SQL dès 1986 par l'ANSI (institut de normalisation américaine), puis par l'ISO (organisation internationale de normalisation), a assuré pour une grande partie le succès du modèle relationnel auprès des entreprises. Fait rare dans le monde informatique, ce langage a été adopté par la quasi-totalité des éditeurs commerciaux qui participent activement à son évolution. SQL est devenu le standard de fait, même si aucun éditeur ne respecte à la lettre la norme. D'ailleurs, à partir de SQL 2, il existe une définition de quatre niveaux de compatibilité avec la norme officielle. La normalisation de ce langage garantit sa pérennité, même si son évolution s'en trouve ralentie. Les requêtes écrites pour un SGBD fonctionnent en général sans trop de modifications avec un autre SGBD, ce qui permet d'envisager des migrations moins douloureuses et de conserver une partie de l'investissement initial.

Figure 1.3

Modèle relationnel.

Ouvrage (Cote, Titre, Auteur,Editeur)

Cote	Titre	Auteur	Editeur
1	La programmation sous Unix	J. M. Rifflet	MC Graw Hill
2	Les bases de données	G. Gardarin	Eyrolles
3	Internet pour les nuls	C. Baroudi	First Interactive
4	Le langage C	C. Delannoy	Eyrolles
5	Petit Larousse illustré	P. Larousse	Larousse
6	Godel, Escher & Bach	D. Hofstadter	Basic Book Inc.

Modèle objet

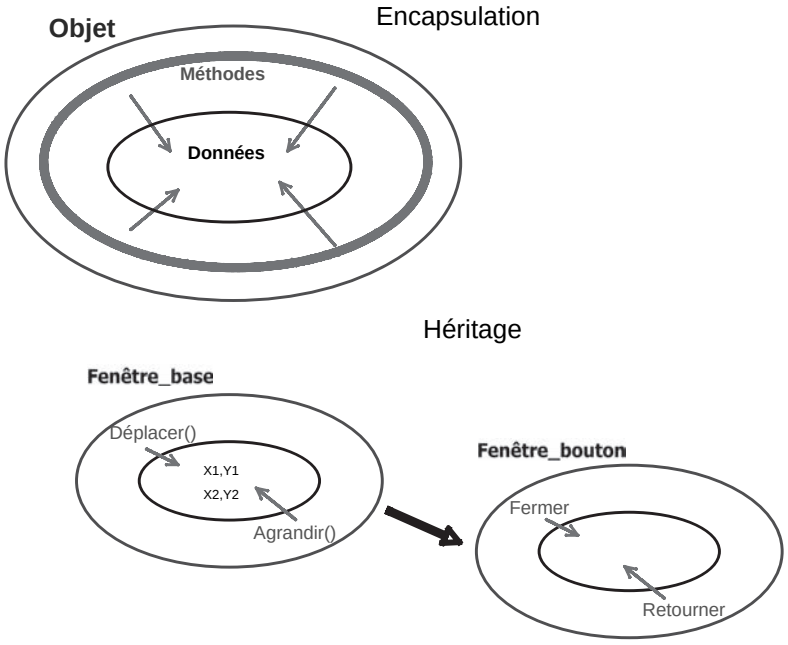
Dans le sillage du développement des langages orientés objet (C++, Java…) dans les années 1980, le **concept objet** a été adapté aux bases de données. Plusieurs raisons, en dehors des qualités reconnues de l'approche objet, ont conduit à définir une extension objet pour les bases de données (voir figure 1.4).

La première est que le modèle relationnel, dans sa simplicité, ne permet pas de modéliser facilement toutes les réalités. La deuxième est qu'un objet permet de représenter directement un élément du monde réel. Les structures d'éléments complexes se retrouvent souvent dispersées entre plusieurs tables dans l'approche relationnelle classique. De plus, le concept objet est mieux adapté pour modéliser des volumes de texte importants ou d'autres types de données multimédias (sons, images, vidéos…). Enfin, il est beaucoup

plus commode de manipuler directement des objets lorsque l'on développe avec un langage à objet (comme C++ ou Java). Les bases de données, on le rappelle, sont dorénavant des briques constitutives des applications. Les **bases de données « orientées objet »** apportent ainsi aux applications développées en langage objet la **persistance** des objets manipulés : ces derniers peuvent ainsi directement être réutilisés par l'application d'origine ou par d'autres sans redéfinition. Ces concepts ont été intégrés à partir de la version 2 de la norme SQL.

Les produits commerciaux adaptés à ces concepts n'ont pas connu une diffusion suffisamment importante. Le monde des bases de données évolue assez lentement : la migration d'un système d'information vers l'objet représente pour une organisation un investissement considérable qui n'est pas toujours justifié. La robustesse et la popularité de l'approche relationnelle, qui a mis presque vingt ans à s'imposer, a également freiné le développement de l'approche objet pure dans les bases de données. Les données modélisées sous forme d'objets sont aussi plus complexes à représenter du point de vue du SGBD et l'on rencontre encore très souvent des problèmes de performance.

Figure 1.4
Modèle objet.



Modèle relationnel-objet

Une demande d'évolution du strict modèle relationnel existe toutefois. En effet, la gestion des données autres que du texte et des nombres – comme des images, du son et des vidéos – implique l'évolution du modèle relationnel. De même, les champs dits « multivalués », disposant de plusieurs valeurs telles qu'une liste de prénoms ou des coordonnées géographiques, ne peuvent pas être modélisés efficacement en utilisant ce type d'approche. L'idée est alors d'intégrer de l'objet au modèle relationnel existant plutôt que d'utiliser l'approche objet pure. Il convient de remarquer que ce type d'évolution a déjà été développé dans le cadre des langages de programmation. Le langage C++ est l'évolution intégrant l'approche objet du langage C et non pas un langage à objet pur comme peut l'être Smalltalk.

Cette extension, adoptée par la plupart des SGBD, se nomme « **relationnel-objet** » et permet aux concepteurs des bases de données de disposer de types évolués « abstraits » plus simples à concevoir et surtout plus commodes à faire évoluer. Elle offre en outre la possi-

bilité de modéliser plus facilement la complexité des organisations (voir figure 1.5). Dans cette optique, la norme SQL a logiquement été adaptée. Dans sa version 3, elle prend en compte l'extension objet. Les types de données sont étendus et les opérations d'encapsulation et d'héritage, typiques de l'approche objet, sont supportées. Cette solution a l'avantage d'offrir un bon niveau de compatibilité avec l'approche précédente très répandue et d'effectuer ainsi une migration plus aisée.

Figure 1.5

Modèle relationnel-objet.

Cote	Titre	Auteur	Editeur
1	Le vide	Nom: Durand, Atamp / Prénom: Dal, Charles	Les éditions du temps qui ne passe pas
2	La vie sans mode d'emploi	Nom: Brassé, Duporche, Château, Paclaire / Prénom: Alexis, Jean-Marie, Romain, Anne-Isabelle	Romazava

2.3 ÉVOLUTION DE L'UTILISATION DES BASES DE DONNÉES

Cette section présente une description de nouvelles manières de stocker ou d'utiliser les bases de données. La différence par rapport à la section précédente est que l'on ne remet pas en cause le modèle utilisé pour décrire les données sauf, dans une certaine mesure, pour le cas de XML.

Base de données réparties

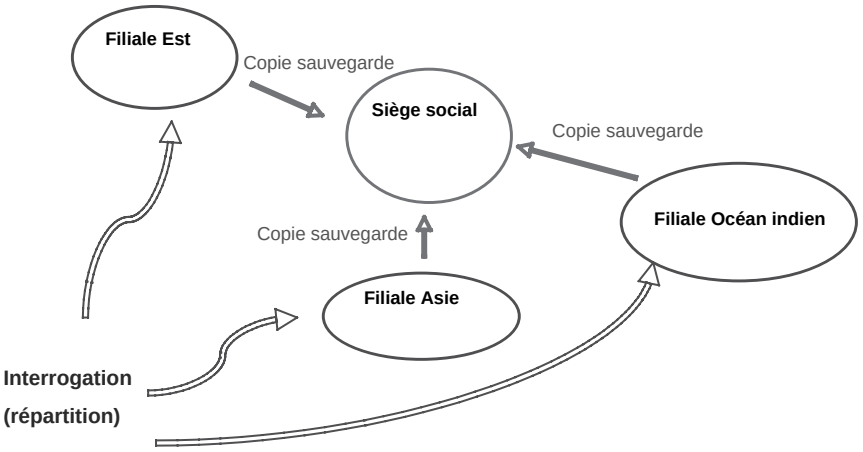
Le déploiement des réseaux ainsi que l'augmentation de leur débit ces dernières années ont conduit à répartir les données sur plusieurs sites géographiques, ce qui facilite la politique de décentralisation des organisations. Ce type d'architecture masque la répartition de l'information tout en garantissant une gestion transparente aux utilisateurs, comme s'ils disposaient d'une seule base de données (voir figure 1.6). Les bases de données réparties assurent ainsi une plus **grande fiabilité**, de **meilleures performances** et facilitent l'**évolution du système d'information**.

- **La fiabilité et la sécurité.** On effectue une copie de sécurité des données sur un site distant à intervalles réguliers pour éviter le désastre de la perte de données due à un incendie par exemple.
- **La disponibilité.** On procède à des réplications quasi permanentes des données dans le but de « rapprocher » les utilisateurs des données d'un point de vue de la topologie du réseau. On améliore également le temps de réponse en répartissant la charge entre les serveurs. Cette distribution est gérée de manière intelligente par les systèmes informatiques, ce qui permet de répartir l'information efficacement sur les différents sites, en fonction des accès utilisateurs. Ces principes sont notamment utilisés par les moteurs de recherche.
- **Les données sont réparties sur des sites séparés.** Le dispositif permet de masquer cet aspect aux utilisateurs et de fonctionner comme si un seul serveur était présent sur un seul site. L'évolution du système est rendue totalement **transparente** pour les utilisateurs : en cas notamment de changement de machine à la suite d'une panne, de

modification de localisation, d'augmentation de la taille de la base, d'ajouts d'ordinateurs sur le réseau afin d'augmenter la capacité de stockage de la base de données.

Figure 1.6

Base de données réparties.



Ces technologies possèdent néanmoins des inconvénients. La sécurité sur les réseaux informatiques nécessite beaucoup plus de travail que dans le cas d'un système non réparti. Les techniques de sécurité à mettre en œuvre sont plus complexes et plus coûteuses.

Extraction d'informations non explicitement stockées dans une base de données

Il existe deux manières de « créer » de la nouvelle information à partir de l'information stockée explicitement dans une base de données. Soit on est capable d'énoncer des règles précises de constitution de l'information à partir du sens même des données, soit on utilise des méthodes d'analyse afin de trouver des corrélations sur un volume de données important, ce qui permet ensuite d'en déduire des règles.

La première possibilité a été formalisée et mise en œuvre sous le nom de **base de données déductives**, dans le but d'utiliser des méthodes semblables à celles pratiquées pour la déduction en intelligence artificielle. On définit un ensemble de règles et on les applique aux données de la base, à l'aide d'un programme que l'on appelle un « moteur d'inférence ». Les SGBD qualifiés de déductifs utilisent à cet effet un dérivé du langage Prolog : **Datalog**.

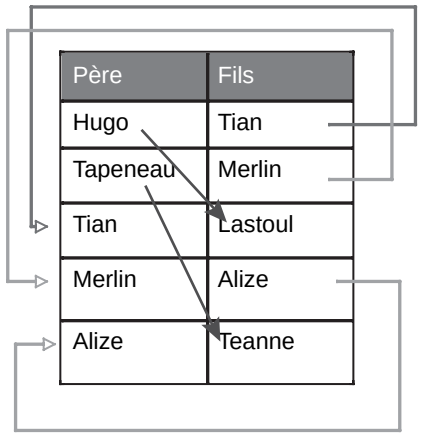
Les relations de parenté entre individus sont une bonne illustration de l'utilisation des bases de données déductives. Intuitivement, on peut appréhender le fonctionnement général de ces SGBD déductifs en considérant l'exemple suivant. On suppose que l'on a une base de données qui représente les relations « père-fils » (voir figure 1.7). Une règle très simple permet de définir un lien de parenté « ancêtre » :

Si père-fils(X,Y) et père-fils(Y,Z), alors ancêtre(X,Z).

La seconde possibilité d'obtenir l'information nouvelle à partir de l'information existante relève des méthodes dites de « **fouilles de données** » (ou **data mining**). Ce type d'exploitation des bases de données a connu un grand succès ces dernières années, par l'analyse de grands volumes de données afin d'identifier des corrélations entre des valeurs de champs. Par exemple, « les personnes moustachues de plus de quarante ans habitant une maison individuelle lisent plutôt des revues de psychologie ». Les techniques d'analyses employées sont une alchimie concoctée à partir de statistiques, de réseaux de neurones, de classifications et autres techniques employées en intelligence artificielle. Une fois ces « règles » établies, on peut les utiliser au sein des bases de données déductives décrites précédemment.

Figure 1.7

Base de données déductives père-fils.



L'information décisionnelle ainsi extraite a de nombreux débouchés : du ciblage marketing à la médecine en passant par la prévision financière. Les méthodes s'affinent et deviennent réellement efficaces, ce qui a cependant donné lieu à quelques escroqueries, les entreprises spécialisées dans le domaine refusant bien sûr de donner les spécifications de leurs méthodes d'analyses, qui relevaient parfois de la divination, au motif de ne pas perdre leur avantage concurrentiel.

Entrepôts de données (datawarehouse)

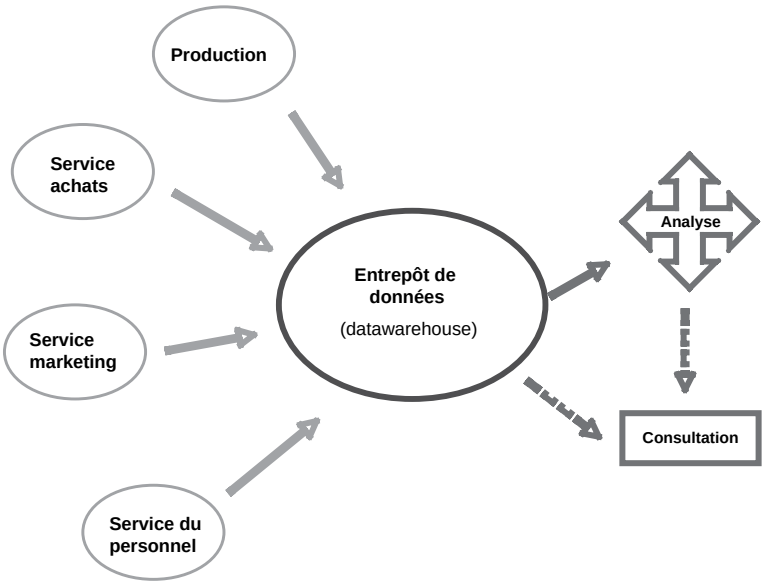
Les entrepôts de données sont des bases de données « récapitulatives », constituées à partir de différentes sources de l'entreprise (comptabilité, ventes, achats, service du personnel…) pour disposer d'un accès homogène à l'ensemble des données. Les données dispersées ne peuvent pas être exploitées directement en tant qu'information décisionnelle. En effet, il n'est pas possible d'analyser efficacement dans le temps des indicateurs de gestion par métiers ou par individus. Les données sont alors rapatriées et stockées dans un **entrepôt de données** ou **datawarehouse** (voir figure 1.8).

Si l'information est disponible, on stocke les différentes valeurs d'une donnée, ce qui permet de conserver un historique dans le temps de certaines données : elles sont dites **historisées**. Ce dernier aspect est désigné parfois aussi comme une **base de données temporelle**, dans la mesure où le SGBD procure des outils d'analyse adaptés. Il est alors possible d'effectuer des analyses décisionnelles qui combinent tous les paramètres de l'entreprise : c'est l'**analyse décisionnelle multidimensionnelle**. Les techniques précédentes de fouille de données et de déduction peuvent être utilisées. Elles sont combinées avec d'autres outils d'analyse de l'évolution d'une donnée à partir de son historique pour en extrapoler des tendances et ainsi faire des prévisions.

L'idée des entrepôts de données est plus qu'intéressante, mais il est évident que sa mise en œuvre est délicate tant d'un point de vue du stockage de l'information (volume de données, hétérogénéité des données à traiter…) que de l'analyse des données (comment analyser, quelle information stocker…). En outre, il s'agit le plus souvent du poste budgétaire le plus onéreux dans un projet d'informatique décisionnelle.

Figure 1.8

Entrepôts de données.



XML

Le *World Wide Web* est la mise en œuvre du concept d'hypertexte par l'utilisation de la structure de communication fournie par Internet. Les fichiers dispersés sur différentes machines du réseau Internet peuvent ainsi être associés. Son succès a provoqué un glissement de l'idée originale, qui consistait à relier simplement des textes entre eux, vers la notion d'interface universelle. On ne transmet plus seulement le contenu d'un fichier statique, mais également le résultat de l'exécution d'un programme : le contenu devient donc dynamique. Par extension, on imagine aisément que le mécanisme du Web peut également transmettre le résultat de l'interrogation d'une base de données.

Le concepteur du Web, T. B. Lee, s'est appuyé pour sa mise en œuvre, outre la structure technique existante d'Internet, sur un langage de description de documents utilisant des balises : le SGML (*Standard Generalized Markup Language*). Le découpage du document par les balises est décrit dans un document associé que chacun peut créer en fonction des ses besoins : la DTD (*Data Type Definition*). Cette dernière, formulée dans un langage normalisé, permettra à des programmes spécialisés, les « parsers », de vérifier si le document est conforme à la structure attendue et d'en extraire facilement le contenu. Le langage SGML est assez complexe à manipuler, car il est très général. Pour des besoins spécifiques, on utilise des versions simplifiées, telles que HTML ou XML.

T. B. Lee a donc développé un langage de présentation, HTML (*Hyper Text Markup Language*), basé sur les notions développées par SGML. Ce langage permet essentiellement de spécifier des critères de présentation (gras, souligné, couleur…) et, bien sûr, de décrire les liens entre les fichiers. Il ne comprend aucun élément de structuration des données du texte contenu dans la page HTML. Les moteurs de recherche parcourent le Web et indexent le contenu des pages par rapport à des listes de mots clés combinés à d'autres méthodes (beaucoup) plus sophistiquées. Cependant, ils ne peuvent différencier dans le texte le titre d'un résumé ou d'une légende associée à une image. L'efficacité et la pertinence de l'indexation en sont diminuées d'autant.

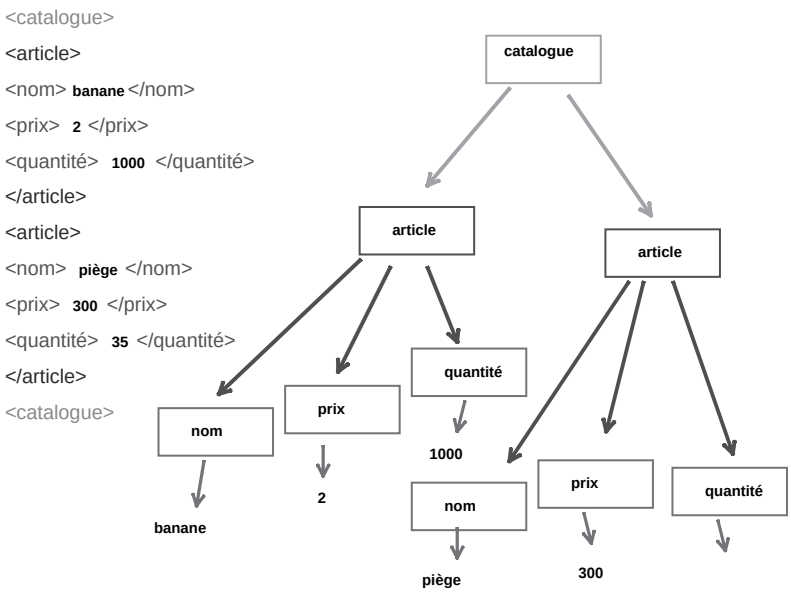
Pour remédier à cela, le W3C (*World Wide Web Consortium*) a défini un langage, qui est également une simplification de SGML, permettant de décrire la structure interne d'un document : XML (*eXtended Markup Language*). La structure d'un document XML est représentée sous la forme d'un arbre tout comme celle d'un document HTML. La description de

cette structure arborescente se trouve dans une DTD ou plus récemment dans un schéma XML. Le schéma est plus souple et permet d'employer les mêmes outils de traitement que pour les fichiers XML. Le langage HTML possède évidemment lui aussi une DTD, mais, à la différence de XML, elle est normalisée par le W3C et ne peut être modifiée et adaptée pour ses besoins propres (voir figure 1.9).

L'objectif à terme est que les fichiers du *World Wide Web* soient désormais décrits en utilisant XML à la place de HTML. La présentation des données repose alors sur les « feuilles de styles » (telles que *eXtended Stylesheet Language*) pour générer les formats de présentation classiques (HTML, PDF, PostScript…) à partir du format XML. De cette manière, les moteurs de recherche pourront extraire directement par exemple le résumé ou le titre d'un paragraphe d'un fichier XML. L'indexation peut ainsi être plus précise ; un mot figurant dans un titre étant plus important que le même mot dans une note de bas de page.

Figure 1.9

Structure arborescente XML.



Quel est le rapport avec les bases de données ? Comme on l'a vu précédemment, une page Web peut être le résultat d'une requête provenant d'un SGBD ; c'est même devenu le moyen le plus courant d'interroger un SGBD. Dans cette optique, si le SGBD est capable de générer directement du XML, cela facilite le processus. C'est d'autant plus vrai que le passage du modèle relationnel à un modèle arborescent de type XML est parfois complexe et qu'il est bien agréable que le SGBD sache le faire.

Le langage de description XML, par sa versatilité, s'impose comme un format d'échange universel. C'est évident pour des fichiers générés par un traitement de texte : on sépare ainsi l'aspect structurel de l'aspect présentation. On se donne également la possibilité d'ouvrir le document indépendamment du logiciel utilisé, ce qui garantit sa pérennité. De la même manière, XML est utilisé comme format d'échange entre SGBD et d'un SGBD vers d'autres logiciels (tableurs, traitements de texte…).

Le langage XML est adopté petit à petit par la plupart des éditeurs et il est amené à jouer un rôle croissant dans les échanges de données. De plus en plus de SGBD sont capables de produire des résultats de requête en XML, d'importer du XML et acceptent même de gérer les données directement dans ce format. Cette dernière possibilité implique que les SGBD supportent des données moins bien structurées : cette capacité constitue l'une des évolutions futures des SGBD.

Contenu multimédia

Le multimédia s'est fortement développé depuis la fin des années 1990. La demande étant très forte dans ce domaine, les éditeurs de SGBD se doivent d'intégrer de l'information multimédia (image, son, vidéo…) dans les bases de données. Les **bases de données multimédias** posent de nouveaux problèmes, en particulier pour effectuer des recherches sur les contenus multimédias, ce qui est par nature difficile.

Une solution est d'effectuer une indexation préliminaire manuelle à l'aide de mots clés qui permettent d'opérer par la suite des interrogations, mais cela semble illusoire de réaliser ce traitement pour des volumes importants de documents multimédias. Dans le cas contraire, il existe des méthodes de recherche sur des fichiers de type image par rapport à des schémas prédéfinis. Cette possibilité reste pour l'instant plutôt du domaine de la recherche, même si l'on est déjà (malheureusement) capable d'identifier des visages par rapport à un modèle dans certaines conditions.Dans le même ordre d'idée, il est déjà possible d'utiliser des techniques pour évaluer le style de musique (classique, jazz, pop…) d'un fichier en analysant son contenu. On est cependant assez loin de pouvoir identifier un genre de film en se basant sur l'analyse des images.

Les bases de données multimédias constituent un sujet de recherche très prometteur et très actif. Ces problématiques relèvent pour l'instant plutôt des préoccupations des sociétés développant les moteurs de recherche que des bases de données au sens strict. Cependant, le groupe de normalisation ISO/IEC prépare l'évolution pour le multimédia de la norme SQL-MM, qui est une évolution pour le multimédia de la norme SQL.

3 Systèmes de gestion de bases de données

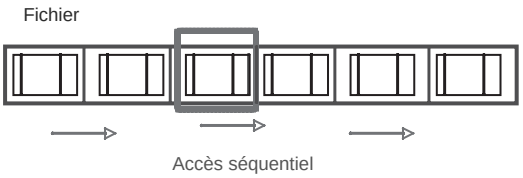
On a beaucoup parlé dans les sections précédentes des SGBD. Un SGBD est un logiciel complexe qui permet de gérer et d'utiliser les données que l'on stocke en utilisant les modèles cités précédemment. La première partie de la section permet de comprendre le mécanisme d'abstraction à partir duquel se fait le passage du simple fichier informatique à la gestion de l'information qui se fonde sur l'utilisation d'un SGBD. Une seconde partie détaille le modèle en couches des SGBD ainsi que leurs fonctionnalités de base.

3.1 Fichiers informatiques

Un fichier peut être vu (historiquement) comme un morceau de bande magnétique. Les mécanismes de gestion de fichiers des langages de programmation, comme le langage C, fonctionnent encore avec cette métaphore. Lorsqu'on utilise un fichier pour stocker de l'information, il est nécessaire de prévoir un découpage de celui-ci par enregistrements, souvent de taille fixe. Pour passer d'un enregistrement à l'autre, il suffit alors d'avancer la « tête » de lecture de la taille d'un enregistrement (voir figure 1.10). Les données sont stockées dans l'enregistrement par un découpage interne suivant la taille de chaque donnée. On utilise généralement des structures de données (par exemple, en langage C) pour récupérer directement chaque valeur de champ dans une variable. Dans une base de données, on recherche les données par leur contenu. Pour retrouver l'une d'entre elles, il faut donc parcourir tous les enregistrements un à un (**recherche séquentielle**), et en examiner le contenu.

Figure 1.10

Fichier informatique.



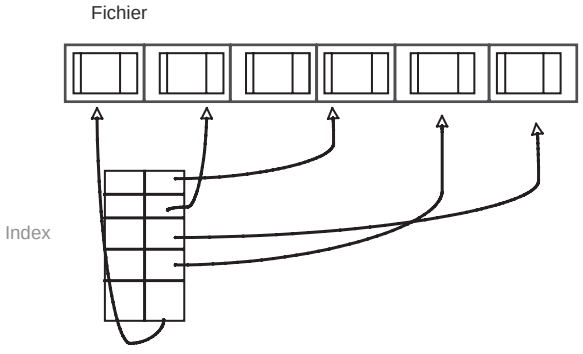
Dans le cas d'un accès séquentiel, la recherche d'un enregistrement en position n nécessite d'accéder aux n–1 enregistrements qui le précèdent. Si l'on recherche l'article de référence DR-NetCard10.102, contenu dans le fichier Article, il sera nécessaire de parcourir tous les enregistrements, depuis le début du fichier jusqu'à l'article recherché. Pour retrouver une information, il faut donc parcourir tous les enregistrements un à un et en examiner le contenu.

Une alternative au parcours séquentiel est de construire des tables descriptives afin d'accélérer l'accès aux données. Une première table permet l'**accès direct** à un enregistrement par une « clé » associée à l'adresse (pointeur) de l'enregistrement. On rappelle que c'est ce mécanisme de « pointeurs » sur des enregistrements qui est modélisé dans les modèles hiérarchiques ou réseaux pour faire le lien entre des enregistrements. On constitue ainsi le graphe qui permet de « naviguer » dans l'ensemble des enregistrements.

Une seconde table contient l'ordre relatif des enregistrements ordonnés suivant les valeurs d'un champ : on appelle cette table un **index** (voir figure 1.11). Cette seconde table permet d'employer des méthodes de recherche par le contenu du champ indexé beaucoup plus efficaces qu'une recherche séquentielle. La recherche dichotomique bien connue est l'une d'entre elles. Une fois l'enregistrement identifié, on y accède directement grâce à la première table. Les techniques de constitution des index constituent un sujet à part entière ainsi que les algorithmes de recherche qui leur sont associés.

Figure 1.11

Fichier et index.



On constate que la simple recherche d'informations recourt déjà à des algorithmes assez sophistiqués et nécessite la construction et le maintien de structures annexes pour être efficace. De même, la destruction d'enregistrements et l'évolution de la structure de la base sont également des opérations lourdes à mettre en œuvre ; elles requièrent souvent la recopie complète des informations dans un nouveau fichier. Cette section nous permet de comprendre pourquoi on utilise préférentiellement des SGBD pour gérer les données. Toutes ces fonctionnalités et bien d'autres que nous allons détailler sont intégrées dans le logiciel.

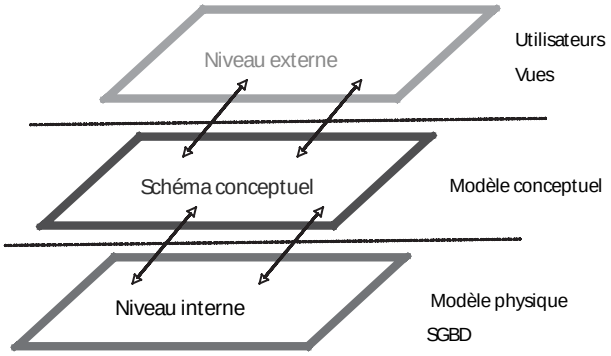
3.2 FONCTIONNALITÉS D'UN SGBD

De même que l'ISO a déterminé un modèle théorique en sept couches pour distinguer les applications réseaux et leurs interactions, il existe désormais un modèle théorique en trois couches (trois niveaux d'abstraction) afin de concevoir et d'organiser les fonctionnalités des SGBD. Ce modèle est élaboré par la commission SPARC de l'ANSI : c'est l'**architecture ANSI/SPARC** (voir figure 1.12). Cette dernière, qui date de 1975, s'inscrit dans les concepts et théories de la première génération des bases de données, dont l'objectif est d'avoir une **indépendance entre les données et les traitements** :

- Niveau **interne** ou **physique**. C'est le niveau le plus « bas ». On décrit les structures de stockage de l'information, ce qui le rend très dépendant du SGBD employé. Il se fonde sur un **modèle de données physique**.
- Niveau **conceptuel**. Il correspond à l'implémentation du **schéma conceptuel de la base de données**, que l'on réalise lors de la phase d'analyse (voir *Modèle Conceptuel des Données*, *Modèle Logique des Données*). Il est utilisé pour décrire les éléments constitutifs de la base de données et les contraintes qui leur sont associées. Il s'agit d'une certaine façon de la « documentation » de la base de données.
- Niveau **externe**. Le niveau externe sert à décrire les **vues** des utilisateurs, c'est-à-dire le **schéma de visualisation des données** qui est différent pour chaque catégorie d'utilisateurs. Un schéma externe permet de masquer la complexité de la base de données complète en fonction des droits ou des besoins des utilisateurs. Cela facilite la lecture et la sécurité de l'information.

Figure 1.12

Niveaux ANSI/ SPARC.



Il s'agit comme pour le modèle réseau de l'ISO d'un cadre de réflexion ; les SGBD ne respectent pas à la lettre le découpage proposé. Ils se doivent cependant de posséder les principales caractéristiques qui découlent de ce modèle en couches :

- **Indépendance physique des données.** Masquer la représentation interne des données ainsi que les méthodes système d'accès aux utilisateurs.
- **Indépendance logique des données.** Permettre la modification du schéma conceptuel des données sans remettre en cause les mécanismes de stockage et de manipulation internes des données.
- **Intégrité des données.** Faire en sorte que l'information résultant des liens entre les données soit cohérente.

Il peut apporter également des fonctionnalités supplémentaires utilisées dans le cadre de bases de données réparties décrites précédemment :

- **Réplication des données.** Copie automatisée de sauvegarde.

- **Virtualisation des données.** Masquage de la distribution géographique des données.
- **Haute disponibilité des données.** Duplication de la base de données sur différents sites pour diminuer la distance client/serveur et la charge des serveurs.

Le but principal de l'utilisation d'un SGBD est de masquer la représentation physique des données et les méthodes d'accès que l'on vient de voir précédemment. Cependant les mécanismes de création, d'indexation et de recherche sous-jacents sont globalement les mêmes. Évidemment, pour des questions d'efficacité, les SGBD utilisent leur propre gestion de fichiers et parfois même contournent le système de fichiers fourni avec le système d'exploitation de la machine.

Un SGBD doit permettre également la manipulation de la structure de la base de données, comme l'ajout et la modification de champs, de manière transparente. Il conserve à cet effet une description de la structure de la base de données que l'on appelle le « dictionnaire de données ». Pour réaliser ces opérations avec l'indépendance souhaitée par rapport à la représentation, le SGBD offre deux langages de haut niveau :

- un *Langage de Description de Données* (LDD) qui permet d'agir sur la structure de la base de données (ajout, suppression et modification des tables) ;
- un *Langage de Manipulation de Données* (LMD) qui permet d'interroger et de mettre à jour le contenu de la base de données.

Ces langages sont de type « non procédural », c'est-à-dire que l'on s'intéresse à l'effet de l'opération (le quoi) et non pas à la manière dont elle est réalisée (le comment). On a pu se rendre compte dans la section précédente du niveau de complexité de certaines opérations qui, grâce à ces langages, sont énoncées simplement. Par exemple, la modification de la taille d'un champ peut être énoncée en une seule instruction avec le LDD. Il est courant que les SGBD modernes implémentent ces langages de manipulation à l'aide d'objets graphiques.

Le SGBD doit également assurer la protection des données en cas de problèmes. Ceux-ci peuvent être la conséquence d'une manipulation malheureuse, mais également d'une panne du système qui survient par exemple à la suite d'une coupure de courant. Dans tous les cas, le SGBD doit permettre de restaurer les données. Ces opérations sont généralement réalisées en utilisant des « journaux » qui enregistrent au fur et à mesure les opérations faites sur la base : c'est le **mécanisme de la journalisation**. Ce journal est utilisé pour refaire, ou défaire le cas échéant, ces opérations.

En ce qui concerne les opérations de modification effectuées sur la base de données, que l'on appelle des **transactions**, des propriétés de mesure de la qualité de ces transactions sont proposées sous le terme ACID :

- **Atomicité.** Une transaction est « atomique » ; elle est exécutée entièrement ou abandonnée.
- **Cohérence.** La transaction doit se faire d'un état cohérent de la base vers un autre état cohérent.
- **Isolement.** Des transactions simultanées ne doivent pas interférer entre elles.
- **Durabilité.** La transaction a des effets permanents même en cas de panne.

À noter que tous les SGBD ne réalisent pas cette propriété ACID pour les transactions.

Les machines sont connectées au réseau ou sont simplement multi-utilisateurs : le SGBD doit permettre de donner l'accès aux bases de données à plusieurs utilisateurs concurremment. **L'accès concurrentiel** implique des opérations algorithmiques complexes à réaliser, puisqu'il faut par exemple empêcher la modification d'une valeur par un utilisateur alors qu'elle est en lecture par un autre. Cela nécessite la gestion d'une structure de description

des utilisateurs comprenant les droits qui leur sont associés pour chaque élément (lecture, modification…) : les **droits d'accès aux données**. Les mécanismes sont les mêmes que ceux qui sont mis en œuvre dans les systèmes d'exploitation multi-utilisateurs.

4 Étapes de la conception des bases de données

On peut décomposer le processus de conception d'une base de données en plusieurs étapes :

- l'analyse du système du monde réel à modéliser ;
- la mise en forme du modèle pour l'intégrer dans un SGBD ;
- la création effective dans le SGBD des structures et leur remplissage (voir figure 1.13).

4.1 Analyse du monde réel

La première étape de la démarche de modélisation des données consiste à effectuer l'analyse de la situation du monde réel à considérer. Cette action s'apparente au travail effectué par une entreprise de consulting. C'est une approche « humaine » qui se fonde en partie sur des entretiens avec les personnels concernés et ressemble plutôt à une analyse du discours et de l'organisation de l'entreprise. C'est lors de cette phase d'analyse que l'on détermine les objectifs du système d'information à concevoir et que l'on identifie tous les éléments à prendre en compte dans le système ; ce sont les champs qui contiendront les données. Un ensemble de champs peut constituer un objet du monde réel. Par exemple les champs « nom », « prénom » et « adresse » que l'on regroupe constituent une « personne ».

Enfin, il faut identifier les liens à modéliser entre ces objets ainsi que les éléments caractéristiques de ces liens. Par exemple une personne achète une voiture à 10 000 euros. Ici les deux objets liés sont « personne » et « voiture », et le prix est le composant du lien. Cette étape est délicate et fondamentale, car elle conditionne l'aspect représentatif et la qualité du modèle du monde réel considéré. Lors de cette phase, il convient également d'exprimer les règles qui définissent le domaine de validité du contenu des champs. Par exemple, le prix d'une voiture ne peut pas être inférieur à 500 euros ou supérieur à 150 000 euros.

Cette modélisation du réel permet de proposer un schéma conceptuel qui servira à la description générale du système d'information. La notion de sens des données et surtout des liens entre les entités ne sera réellement exprimée que dans ce schéma qui est plus proche du monde réel. Ce schéma est souvent réalisé à l'aide de la symbolique du modèle « entité-association » ou, plus couramment aujourd'hui, exprimé avec le langage UML (*Unified Modeling Language*). Il existe différentes méthodes intégrant les concepts présentés ci-dessus. L'objectif est de guider le travail d'analyse et d'aider à la réalisation d'un modèle de données le plus juste possible. Parmi celles-ci, la méthode Merise a connu un certain succès dans le domaine en France.

4.2 Passage au SGBD

La représentation précédente doit être transformée pour la rendre acceptable par le SGBD, qu'il soit relationnel, objet ou relationnel-objet. Souvent, cette étape modifie considérablement les objets du monde réel ainsi que les liens définis dans le schéma précédent. C'est lors de cette phase que l'on vérifie la qualité de la base de données en utilisant les critères

vus précédemment, comme l'élimination de la redondance. Le modèle relationnel procure à cette fin des outils capables de tester la cohérence du système et de le modifier le cas échéant : ce sont les « formes normales », qui seront vues au chapitre 3.

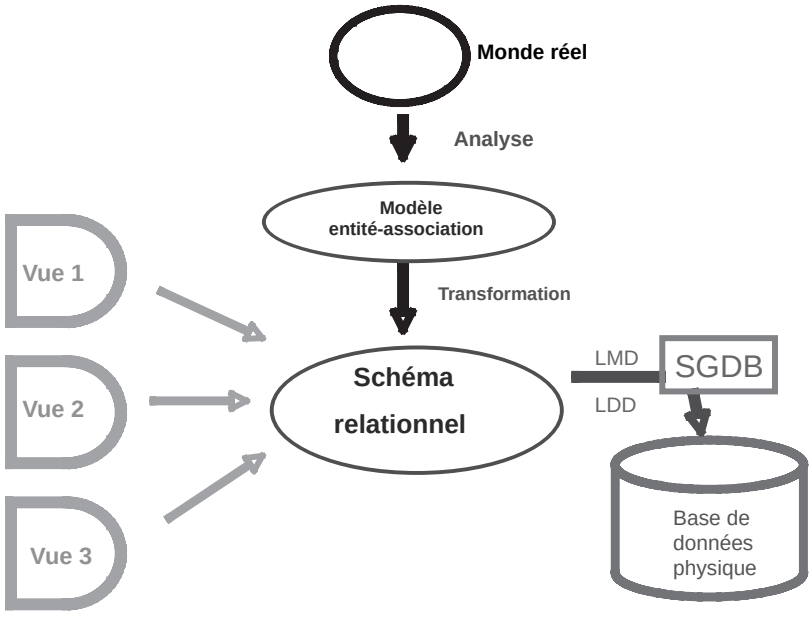
Il est possible de constater des incohérences à ce niveau de l'analyse, ce qui implique de modifier le modèle conceptuel de données développé à l'étape précédente. On obtient un schéma des données qui fournira aux utilisateurs les informations nécessaires pour effectuer leurs requêtes, par exemple la description des noms de tables, de champs et leurs types. Par contre, on perd à ce niveau l'information du « sens » des données et du lien entre elles. Ce schéma n'est guère utilisable en pratique sans le précédent. En effet, comment savoir que les personnes achètent des voitures et non pas le vendeur si l'on ne dispose pas de l'information de liaison entre les objets du monde réel ? C'est également lors de cette phase que l'on définit les « vues » du système d'information qui sont adaptées à chaque catégorie d'utilisateurs.

4.3 Création et utilisation de la base de données

Une fois le schéma précédent défini, on utilise le SGBD pour passer à la création des tables qui constituent la base de données. Puis, on insère évidemment les valeurs dans les tables. Le cas échéant, on crée les vues définies à l'étape précédente et par là même les utilisateurs concernés. Le système est alors opérationnel. Toute cette étape se fait forcément en utilisant le SGBD, alors que les précédentes étaient plus théoriques. La création des tables et l'utilisation de la base de données nécessiteront le langage SQL. Cependant, il existe de nos jours de nombreux outils graphiques dans les SGBD qui masquent l'utilisation du SQL.

Figure 1.13

Étapes de la conception d'une base de données.



5 « Métiers » des bases de données

Comme on peut le constater lorsque l'on considère les différentes étapes de la conception d'une base de données, des acteurs aux compétences très diverses interviennent dans ce processus.

5.1 Consultants/analystes

Ils prennent en charge la première étape qui consiste en l'analyse des activités et des flux d'information mis en jeu dans le monde réel à modéliser. Le profil de ces acteurs n'est pas toujours purement technique, puisque cette phase nécessite parfois beaucoup de dialogues et de psychologie pour parvenir à faire exprimer leurs besoins réels par les futurs utilisateurs. La gageure est de parvenir à faire exprimer correctement les besoins d'informatisation par les utilisateurs du système d'information, afin de proposer un modèle conceptuel de données le plus juste possible.

5.2 Concepteurs de la base

Ce sont les personnes qui s'occupent de traduire le modèle précédent en un modèle logique exploitable par le SGBD. Le concepteur est un spécialiste des bases de données qui prépare les tables, les vues, les schémas d'accès. C'est lui qui renseigne les utilisateurs et programmeurs pour la définition des requêtes. Il n'a pas, en principe, à être spécialisé sur un SGBD particulier, mais en pratique les éléments qu'il manipule sont liés au SGBD qui sera employé. C'est ordinairement lui qui crée les éléments nécessaires à la base de données (tables, vues…) en collaboration avec l'administrateur de la base. C'est parfois la même personne qui est en charge de la partie analyse et de la conception, ce qui peut induire une vision un peu trop orientée techniquement – comme celle d'un programmeur qui écrirait le cahier des charges d'une application. Par contre, le concepteur peut aussi être administrateur du SGBD, ce qui ne pose pas de problèmes particuliers d'approche.

5.3 Administrateurs de base de données (DBA, DataBase Administrator)

L'administrateur a la responsabilité du fonctionnement général du SGBD. Il crée les ressources (bases, comptes) à la demande. Il donne les droits d'accès et gère les personnes qui accèdent au système. Il vérifie que les ressources sont suffisantes (taille du disque, puissance de la machine), effectue les sauvegardes, vérifie les failles de sécurité. Pour ces opérations, il est en relation avec l'administrateur système et réseau de la structure. Ce métier est extrêmement lié au SGBD employé. Il n'y a pas vraiment de normalisation pour les opérations d'administration des SGBD qui sont spécifiques au SGBD et à la version utilisés.

5.4 Utilisateurs standard et programmeurs d'applications

Ce sont eux qui utilisent le système d'information. Ils y ont accès grâce aux vues définies par le concepteur de la base. Ils utilisent les schémas déterminés aux deux premières étapes de la conception. Ils n'ont pas besoin théoriquement d'être spécialisés sur le SGBD employé. En pratique il est préférable, surtout pour les développeurs d'applications, d'avoir de bonnes connaissances du fonctionnement du SGBD. Par exemple, pour optimiser les performances, la manière d'écrire les requêtes peut être assez différente suivant le SGBD employé.

Résumé

Une base de données désigne l'ensemble des données stockées. Pour manipuler ces données, on utilise un SGBD (Système de Gestion de Bases de Données) qui est un logiciel complexe. L'ensemble composé par les données et le SGBD constitue un système d'information. La conception d'une base de données – de la modélisation du monde réel à son implémentation dans le SGBD – fait appel à des techniques et des méthodes très différentes pour chaque étape. Des métiers spécifiques se sont donc développés autour de ces concepts et les mettent en œuvre. Par exemple, l'approche du monde réel s'apparente à l'analyse faite par un cabinet de consulting alors que l'implémentation dans le SGBD et son administration sont proches des métiers informatiques.

Les SGBD ont évolué parallèlement aux concepts de modélisation des bases de données. On est passé d'une organisation comparable à celle des fichiers informatiques (modèles hiérarchiques ou réseaux) à un modèle plus abstrait : le modèle relationnel. Ce modèle est toujours le plus utilisé actuellement. Il est associé étroitement à SQL, un langage normalisé de description, de manipulation et d'interrogation de données. La modélisation objet, adaptée aux bases de données, n'a pas connu un développement considérable, et ce, malgré les avantages qu'elle procure par rapport au modèle relationnel – en particulier pour le typage des données. Comme c'est le cas dans le domaine de la programmation, une approche mixte semble prendre de l'ampleur : le modèle relationnel-objet. Il s'agit d'apporter au modèle relationnel les possibilités étendues de modélisation procurées par les objets sans remettre profondément en question l'existant.

Le développement des réseaux apporte d'autres manières d'utiliser les bases de données, comme la répartition des données pour améliorer leur disponibilité et leur sécurité. L'interfaçage avec le World Wide Web a introduit la prise en compte du langage XML comme format d'échange et de stockage par les SGBD. De nouvelles formes d'interrogation, telles que la « fouille de données » (ou data mining) et les bases de données déductives, permettent d'extrapoler de l'information non explicitement stockée dans les bases de données. Ces approches ainsi que la prise en compte des données multimédias vont faire évoluer les modèles de bases de données et les SGBD que l'on utilise actuellement. Cela se fera probablement sans remettre totalement en cause le modèle relationnel, mais plutôt en le faisant évoluer progressivement.

2 Opérations du modèle relationnel

La manipulation des données dans le modèle relationnel se fait à l'aide d'opérations formelles reposant sur des concepts mathématiques issus de la théorie des ensembles : c'est **l'algèbre relationnelle**. Les opérations de l'algèbre relationnelle portent sur une ou plusieurs relations (ou tables). Le résultat retourné par ces opérations est toujours une relation (ou table).

Cette section présente une liste non exhaustive des différentes opérations de l'algèbre relationnelle. Ces dernières peuvent être regroupées en plusieurs catégories :

- les opérations classiques issues de la théorie des ensembles (union, intersection, différence) ;
- les opérations plus spécifiques du monde relationnel (projection, sélection, jointure) qui constituent les opérations fondamentales de l'algèbre relationnelle ;
- les opérations de types calcul et agrégats qui ne constituent pas réellement des opérations fondamentales (ni ensemblistes, ni relationnelles) mais qui sont le complément indispensable des précédentes.

2.1 Opérations ensemblistes

L'algèbre relationnelle emprunte un certain nombre de concepts à la théorie des ensembles. À l'exception du produit cartésien, ces opérations ont la particularité de ne pouvoir s'utiliser que pour des relations qui ont exactement la même structure ou des structures compatibles. Elles sont toutes de type binaire, car elles s'appliquent à deux relations.

Union

L'opération d'union consiste en la mise en commun des enregistrements de chaque relation. Les enregistrements identiques ne sont intégrés qu'une seule fois. Dans l'exemple ci-après (voir figures 3.3 et 3.4), le tuple dont la valeur du champ 'Numero_carte' vaut 3 ne sera pas dupliqué. L'union est représentée par le caractère « ∪ ». Cette opération sert typiquement à la « consolidation » de données de même type provenant de différentes sources.

Figure 3.3

Relations 'Lecteur_1' et 'Lecteur_2'.

Lecteur_1(Numero_carte, Nom, Age, Ville, Etablissement)

Numero_carte	Nom	Age	Ville	Etablissement
1	Henri	10	Paris	Université Sorbonne
2	Stanislas	34	Paris	Université Jussieu
3	Henriette	44	Lyon	CHU Bron

Lecteur_2(Numero_carte, Nom, Age, Ville, Etablissement)

Numero_carte	Nom	Age	Ville	Etablissement
3	Henriette	44	Lyon	CHU Bron
4	Dominique	19	Nancy	Université Poincaré
5	Isabelle	56	Nancy	INPL

Figure 3.4 **Lecteur_1 ∪ Lecteur_2**

Union des relations 'Lecteur_1' et 'Lecteur_2'.

Numero_carte	Nom	Age	Ville	Etablissement
1	Henri	10	Paris	Université Sorbonne
2	Stanislas	34	Paris	Université Jussieu
3	Henriette	44	Lyon	CHU Bron
4	Dominique	19	Nancy	Université Poincaré
5	Isabelle	56	Nancy	INPL

Différence

L'opération différence consiste à désigner les enregistrements qui appartiennent à une relation sans appartenir à l'autre. La différence est représentée par le caractère « – » (voir figure 3.5). Attention, cette opération n'est pas symétrique ; le résultat de l'opération 'Lecteur_1 – Lecteur_2' est en général différent de 'Lecteur_2 – Lecteur_1'. Elle permet par exemple d'éliminer des enregistrements d'une relation par rapport à une liste.

Figure 3.5 **Lecteur_1 – Lecteur_2**

Différence des relations 'Lecteur_1' et 'Lecteur_2'.

Numero_carte	Nom	Age	Ville	Etablissement
1	Henri	10	Paris	Université Sorbonne
2	Stanislas	34	Paris	Université Jussieu

Intersection

L'opération intersection peut se déduire de la précédente ; elle désigne les enregistrements qui sont communs aux deux relations. L'intersection est représentée par le caractère « ∩ » (voir figure 3.6). Elle permet de trouver les éléments communs à deux relations.

Figure 3.6 **Lecteur_1 ∩ Lecteur_2**

Intersection des relations 'Lecteur_1' et 'Lecteur_2'.

Numero_carte	Nom	Age	Ville	Etablissement
3	Henriette	44	Lyon	CHU Bron

Produit cartésien

Le produit cartésien permet la combinaison des enregistrements de deux relations sans tenir aucun compte du contenu des données. Les relations n'ont donc pas besoin d'avoir la même structure. Le caractère représentant le produit cartésien est « X ».

Figure 3.7 **Relations 'ma_cuisine' et 'musicien'.**

ma_cuisine(Appareil, Couleur)

Appareil	Couleur
Réfrigérateur	rouge
Robot	mauve
Cuisinière	jaune

musicien(Nom, Instrument)

Nom	Instrument
Jaco Pastorius	Basse électrique
Bill Evans	Piano

L'exemple a été choisi de manière à montrer que les relations ne nécessitent pas de rapports entre elles pour faire un produit cartésien. Combiner des appareils de cuisine et des musiciens n'a aucun sens dans la réalité (voir figures 3.7 et 3.8).

Figure 3.8 **Produit cartésien des relations 'ma_cuisine' et 'musicien'.**

Le_resultat(Appareil, Couleur, Nom, Instrument) = ma_cuisine(Appareil, Couleur) X musicien(Nom, Instrument)

Appareil	Couleur	Nom	Instrument
réfrigérateur	rouge	Jaco Pastorius	Basse électrique
réfrigérateur	rouge	Bill Evans	Piano
robot	mauve	Jaco Pastorius	Basse électrique
robot	mauve	Bill Evans	Piano
cuisinière	jaune	Jaco Pastorius	Basse électrique
cuisinière	jaune	Bill Evans	Piano

2.2 Opérations relationnelles

Projection

La projection consiste à extraire certains champs de la relation, ce qui donne à cette dernière un degré inférieur à la relation de départ. Voici la relation obtenue à partir de la relation 'Lecteur' en projetant les champs 'Nom' et 'Ville' (voir figure 3.9).

Figure 3.9 **Projection des champs 'Nom' et 'Ville' de la relation 'Lecteur'.**

Lecteur_proj(Nom, Ville)

Nom	Ville
Henri	Paris
Stanislas	Paris

Nom	Ville
Henriette	Lyon
Dominique	Nancy
Isabelle	Nancy
Olivier	Marseille
Henri	Paris
Jerome	Nancy
Laurence	Bordeaux
Christian	Paris
Antoine	Marseille
Laurence	Paris

Intuitivement, dans une représentation de type table, on conserve uniquement les colonnes sur lesquelles la projection est faite.

Sélection ou restriction

La sélection consiste à extraire les enregistrements de la relation. On utilise des critères pour caractériser les enregistrements sélectionnés. Voici la relation obtenue à partir de la relation 'Lecteur 'en sélectionnant les enregistrements dont le contenu du champ 'Ville' est 'Marseille' (voir figure 3.10). La structure de la relation résultat est la même que celle de la relation de départ.

Figure 3.10 Sélection sur la relation 'Lecteur'.

Lecteur_sel(Numero_carte, Nom, Age, Ville, Etablissement)

Numero_carte	Nom	Age	Ville	Etablissement
6	Olivier	51	Marseille	Université Saint Charles
11	Antoine	16	Marseille	Université Saint Charles

Intuitivement, dans une représentation de type table, on conserve uniquement les lignes répondant au critère.

Jointure

La jointure est l'opération fondamentale de l'algèbre relationnelle qui permettra d'exprimer le sens du lien entre les relations dans le monde réel. La liaison entre les relations s'effectue par le contenu commun d'un champ. L'opération de jointure peut être vue comme une sélection des enregistrements obtenus par le produit cartésien des relations, dont les contenus du champ sur lequel on effectue la jointure sont égaux. On l'appelle dans ce cas une **équijointure**. Les champs dont on compare les contenus sont nommés **champs de jointure**.

On considère les deux relations Lecteur_bis(Numéro_carte, Nom, Num_Etablissement) et Etablissement(Num_Etablissement, Ville, Nom_Etablissement) dont le contenu suit (voir figure 3.11).

Figure 3.11 **Lecteur_bis(Numero_carte, Nom, Num_Etablissement)**

Relations 'Lecteur_bis' et 'Etablissement'.

Numero_carte	Nom	Num_Etablissement
1	Henri	1
2	Stanislas	2
3	Henriette	1

Etablissement(Num_Etablissement, Ville, Nom_Etablissement)

Num_Etablissement	Ville	Nom_Etablissement
1	Paris	Université Jussieu
2	Lyon	CHU Bron
3	Nancy	Université Poincaré
4	Paris	Université Sorbonne

Même si l'on ne dispose pas du modèle conceptuel associé, on constate que l'on peut relier ces deux relations par le champ 'Num_Etablissement'. Les informations concernant l'établissement de la relation Lecteur_bis sont stockées dans la relation Etablissement dont la clé est le champ Num_etablissement. Pour obtenir la liste des lecteurs ainsi que les informations concernant leur établissement, on effectue une jointure entre ces relations sur le champ 'Num_Etablissement' (voir figure 3.12).

Figure 3.12 **Lecteur_joint_Etablissement(Numero_carte, Nom, Num_Etablissement_1, Num_Etablissement_2, Ville, Nom_Etablissement)**

Jointure des relations 'Lecteur_bis' et 'Etablissement' sur le champ 'Num _Etablissement'.

Numero _carte	Nom	Num_Etablis-sement_1	Num_Etablis-sement_2	Ville	Nom _Etablissement
1	Henri	1	1	Paris	Université Jussieu
2	Stanislas	2	2	Lyon	CHU Bron
3	Henriette	1	1	Paris	Université Jussieu

Le champ 'Num_Etablissement' y figure deux fois car une occurrence vient de la relation 'Lecteur' et l'autre de la relation 'Etablissement'. Afin de ne conserver qu'une valeur du champ 'Num_Etablissement', on utilise l'opération de **jointure naturelle** (voir figure 3.13).

Figure 3.13 **Lecteur_jointnat_Etablissement(Numero_carte, Nom, Num_Etablissement, Ville, Nom_Etablissement)**

Jointure naturelle des relations 'Lecteur_bis' et 'Etablissement' sur le champ 'Num _Etablissement'.

Numero_carte	Nom	Num _Etablissement	Ville	Nom _Etablissement
1	Henri	1	Paris	Université Jussieu
2	Stanislas	2	Lyon	CHU Bron
3	Henriette	1	Paris	Université Jussieu

On peut considérer l'opération de jointure comme une sélection sur le produit cartésien des deux relations. On ne conserve que les lignes dont le contenu du champ sur lequel s'effectue la jointure est égal. Les lignes grisées sont celles qui sont sélectionnées lors d'une jointure (voir figure 3.14).

Figure 3.14

Produit cartésien des relations 'Lecteur_bis' et 'Etablissement'.

Produit_cartesien_Etablissement(Numero_carte, Nom, Num_Etablissement_1, Num_Etablissement_2, Ville, Nom_Etablissement)

Numero _carte	Nom	Num_Etablis-sement_1	Num _Etablissement	Ville	Num_Etablis-sement_2
1	Henri	1	1	Paris	Université Jussieu
1	Henri	1	2	Lyon	CHU Bron
1	Henri	1	3	Nancy	Université Poincaré
1	Henri	1	4	Paris	Université Sorbonne
2	Stanislas	2	1	Paris	Université Jussieu
2	Stanislas	2	2	Lyon	CHU Bron
2	Stanislas	2	3	Nancy	Université Poincaré
2	Stanislas	2	4	Paris	Université Sorbonne
3	Henriette	1	1	Paris	Université Jussieu
3	Henriette	1	2	Lyon	CHU Bron
3	Henriette	1	3	Nancy	Université Poincaré
3	Henriette	1	4	Paris	Université Sorbonne

On verra au chapitre consacré à SQL que c'est l'une des manières d'écrire une jointure.

Jointure externe

La jointure externe n'est pas réellement une opération de base de l'algèbre relationnelle. Elle est cependant nécessaire pour pouvoir répondre plus facilement à des questions du type « Quels sont les établissements qui n'ont pas de lecteurs ? » Il s'agit d'une opération de jointure étendue qui inclut dans le résultat les lignes n'ayant pas de correspondance sur le contenu du champ de jointure. Voici le résultat de la jointure externe de la relation Etablissement avec la relation Lecteur sur le champ 'Num_Etablissement' (voir figure 3.15). On met en correspondance les valeurs du champ 'Num_Etablissement' de toutes les lignes de la relation 'Etablissement' avec celles de la relation 'Lecteur'.

Figure 3.15

Jointure externe des relations 'Lecteur_bis' et 'Établissement' sur le champ 'Num _Etablissement'.

Etablissement_jointext_ Lecteur(Num_Etablissement_1, Ville, Nom_Etablissement, Numero_carte, Nom, Num_Etablissement_2)

Num_Etablissement_1	Ville	Nom _Etablissement	Numero _carte	Nom	Num_Etablissement_2
1	Paris	Université Jussieu	1	Henri	1
2	Lyon	CHU Bron	2	Stanislas	2
3	Nancy	Université Poincaré	NULL	NULL	NULL
4	Paris	Université Sorbonne	NULL	NULL	NULL

Ici, les valeurs 3 et 4 du champ 'Num_Etablissement_1' provenant de la relation Etablissement n'ont pas de correspondance dans la relation 'Lecteur'. Les lignes sont tout de même incluses dans le résultat mais les champs provenant de la relation 'Lecteur' prennent la valeur 'NULL'. Cette valeur signifie que le champ ne contient pas de valeur.

Pour répondre à la question « Quels sont les établissements qui n'ont pas de lecteurs ? », il nous suffit de sélectionner les lignes qui contiennent la valeur 'NULL', par exemple dans le champ 'Numéro_carte'. Dans un second temps, on effectue une projection sur le champ 'Nom_Etablissement' (voir figure 3.16).

Figure 3.16

Sélection et projection sur la jointure externe des relations 'Lecteur_bis' et 'Etablissement' sur le champ 'Num _Etablissement'.

Etablissement_jointext_Lecteur_selproj(Nom_Etablissement)

Nom _Etablissement
Université Poincaré
Université Sorbonne

La jointure externe n'est pas une opération **symétrique**. L'opération inverse, c'est-à-dire la jointure externe de la relation 'Lecteur' avec la relation 'Etablissement' sur le champ 'Num_Etablissement' donne dans ce cas le même résultat qu'une jointure naturelle. En effet, toutes les valeurs du champ 'Num_Etablissement' de la relation 'Lecteur' ont une correspondance dans la relation 'Etablissement'. C'est ce que l'on souhaite puisque la relation 'Etablissement' fait office de relation de référence pour le champ 'Num_Etablissement' de la relation 'Lecteur'. L'opération de jointure externe peut être utilisée pour détecter ce type d'incohérence.

2.3 Calculs et agrégats

Une règle dans le domaine des bases de données est que tout ce qui peut se calculer ne doit pas être stocké. On évite ainsi la perte de place et l'incohérence qui peut en découler suite au stockage d'informations redondantes. Les opérations de l'algèbre relationnelle présentées dans les sections précédentes ne permettent pas de créer de nouveau champ par calcul à partir du contenu d'autres champs sans utiliser un langage de programmation. Des

fonctions de calculs ont été définies afin de répondre à ce besoin. Elles seront détaillées au chapitre sur « SQL ».

On considère la relation La_boutique(Num_facture, Article, Prix, Quantite) [voir figure 3.17].

Figure 3.17 **La_boutique(Num_Facture, Article, Prix, Quantite)**

Relation 'La_boutique'.

Num_Facture	Article	Prix	Quantite
101	Bananes	12	45
1034	Choux	5	129
2345	Riz	4	60
0987	Gazelle	15	48

On suppose que l'on veut exemple trouver le total pour chaque facture en multipliant le prix par la quantité. Il est alors possible d'ajouter un champ 'Total' dont le contenu sera calculé par l'expression 'Prix' * 'Quantité' (voir figure 3.18).

Figure 3.18 **La_boutique_total(Num_FactureArticle, Prix, Quantite, Total)**

Relation 'La_boutique' avec le champ calculé 'Total'.

Num_Facture	Article	Prix	Quantite	Total
101	Bananes	12	45	540
1034	Choux	5	129	645
2345	Riz	4	60	240
0987	Gazelle	15	48	720

Il est également possible d'effectuer des opérations statistiques globales d'un champ, comme les calculs du nombre d'enregistrements, de moyenne, de maximum, de minimum. On obtient dans ce cas une nouvelle relation réduite à une ligne et une colonne qui contient la valeur calculée. En effet, le résultat d'une opération sur une relation est toujours une relation (voir figure 3.19).

Figure 3.19 **La_boutique_moyenne(Moyenne_Prix)**

Moyenne des prix de la relation 'La_boutique'.

Moyenne_Prix
9

De même, les opérations de base de l'algèbre relationnelle ne permettent pas de répondre à des questions du type : « Combien trouve-t-on de personnes par ville ? » La solution consiste alors à regrouper les enregistrements qui contiennent les mêmes valeurs dans le champ 'Ville' : ce regroupement s'appelle un **agrégat**. On applique ensuite à chaque sous-relation ainsi constituée une opération statistique, dans notre cas l'opération de comptage (voir figure 3.20).

Figure 3.20 **Lecteur_parville(Ville, Nombre)**

Agrégat par ville de la relation 'La_boutique'.

Ville	Nombre
Bordeaux	1
Lyon	1

Ville	Nombre
Marseille	2
Nancy	3
Paris	5

3 Passage du modèle conceptuel au relationnel

On a exposé au chapitre sur l'analyse du monde réel deux formalismes différents – entité-association et UML – pour représenter le modèle conceptuel obtenu. Il existe des règles simples pour passer de ces modèles à un ensemble de relations qui sera utilisable directement dans le SGBD. Cet ensemble de relations s'appelle le **schéma relationnel** et constitue le **modèle logique des données**. On présente dans cette section les règles de transition qui permettent d'exprimer le schéma relationnel à partir du modèle conceptuel des données. Pour illustrer ces règles, on utilise la terminologie du modèle entité-association, mais leur expression est aisément adaptable au formalisme UML.

3.1 Règles générales

Deux grandes règles générales s'appliquent toujours, quel que soit le cas. Les exceptions que l'on aborde ensuite permettent simplement d'obtenir un schéma plus compact :

- Une entité devient une relation composée des champs de l'entité. La clé de cette relation est la même que celle de l'entité.
- Une association devient une relation composée des deux clés des entités associées. L'ensemble de ces deux clés constitue celle de la relation. Lorsqu'elle en possède, les champs de l'association forment les champs « non clés » de cette nouvelle relation.

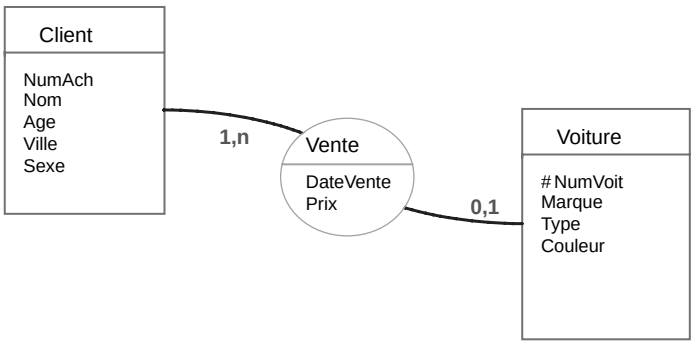


Figure 3.21
Modèle entité-association de la base de données 'casse'.

À partir du modèle entité-association de la base de données 'casse' élaboré au chapitre précédent (voir figure 3.21), on obtient trois relations en appliquant ces règles. Les entités 'voiture' et 'personne' deviennent les relations voiture(NumVoit, Marque, Type, Couleur) et personne(NumAch, Nom, Age, Ville, Sexe). L'association se transforme en relation vente(DateVent, Prix, NumAch, NumVoit).

Préservation des données

Ce chapitre traite de la sécurité des données qui est une notion fondamentale des bases de données. En effet, le pire pour une base de données est la perte ou la modification de données. Des problèmes sérieux peuvent être provoqués de manière intentionnelle ou accidentelle, mais le résultat obtenu est le même.
L'activité des organisations repose sur les données : il convient par conséquent de les protéger, mais également d'en assurer la disponibilité permanente.
On distinguera plusieurs niveaux de granularité quant aux recommandations générales concernant la sécurité des données :

- la protection de l'accès à la machine d'un point de vue physique ou réseau, les aspects de durée de vie du système ainsi que la politique de sauvegarde ;
- la politique de restriction d'accès aux données du SGBD par des comptes et l'utilisation des vues ;
- la capacité des outils du SGBD à protéger les opérations sur les données, à les restaurer et à revenir en arrière, comme pour les transactions.

On introduit en fin de chapitre un outil complémentaire : le « trigger ». Les triggers (ou déclencheurs) sont utilisés pour forcer une bonne gestion du contenu des données lors des opérations d'insertion, de mise à jour ou de suppression de données.

1 Contrôle d'accès et sauvegarde

Toute réflexion autour de la sécurité doit se concevoir de manière globale en utilisant en premier lieu le bon sens : il est inutile d'installer une porte blindée inviolable sur une palis-sade en bois. Les recommandations qui sont énoncées dans cette section sont applicables à tous les systèmes informatiques, en particulier à ceux qui sont connectés à un réseau ou situés dans un milieu qualifié d'« hostile ».

On décrit ici plusieurs types de préoccupations :

- l'accès physique à la machine ;
- l'accès distant à la machine à travers le réseau ;
- les aspects de pérennité de l'ensemble machine, système d'exploitation et SGBD ;
- les sauvegardes, la redondance des données et leur accessibilité.

1.1 Accès physique à la machine

Cette mesure de sécurité informatique élémentaire est très souvent négligée. Procurer un accès physique à la machine permet en général de contourner toutes les protections liées au système d'exploitation ou au SGBD que l'on aborde dans ce chapitre. Il suffit de démarrer une machine avec un autre système d'exploitation présent sur un CD ou une clé USB par exemple pour obtenir un accès au disque et ainsi aux données associées au SGBD. La première précaution à prévoir consiste à protéger le BIOS (*Basic Input Output System*) ou l'équivalent de la machine pour l'empêcher de démarrer sur un autre disque que celui qui contient le « bon » système d'exploitation.

Le fait de pouvoir accéder directement à la machine permet également à une personne animée de mauvaises intentions d'empêcher l'accès aux données en détruisant physique-ment le disque sur lequel se trouvaient les fichiers ou même la machine.

Par conséquent, la ou les machines qui contiennent le SGBD et les fichiers de données doi-vent se trouver dans des pièces dont l'accès est, au minimum, contrôlé. Dans le même ordre d'idée, l'alimentation électrique ne doit pas être accessible. On a tous entendu ou vécu une histoire de machines ou d'éléments réseaux débranchés par erreur pour effectuer le ménage, sans parler de la malveillance. Comme on peut le constater, les précautions concernant l'accès physique à la machine ne sont pas toujours d'ordre purement technique ; elles relèvent souvent du simple bon sens commun.

1.2 Accès à la machine par le réseau

L'accès à la machine par le réseau est devenu une porte d'entrée privilégiée pour les per-sonnes malintentionnées, car rares sont les serveurs qui ne sont pas connectés aujourd'hui. On distingue ainsi principalement deux aspects de problèmes de sécurité :

- les attaques « internes » qui proviennent de personnes ayant accès à la machine par un compte standard associé à un mot de passe correct ;
- les attaques « externes » qui se font en profitant d'une faille du système d'exploitation ou d'une des applications installées sur la machine.

Dans les deux cas, on peut être confronté à un problème de destruction de données par simple effacement de fichiers de données. Cette opération d'effacement sera rendue plus ou moins facile suivant les systèmes employés. Dans le premier cas, il est en principe

impossible qu'un simple utilisateur efface les fichiers des autres, mais certains systèmes sont plus « perméables » que d'autres. Dans le second cas, si l'on peut devenir administrateur sur la machine concernée du point de vue du système d'exploitation, tout est possible. S'il s'agit d'un cas de malveillance, les dégâts peuvent être considérables.

Les protections dans ce domaine ne dépendent pas spécifiquement de l'administrateur du SGBD, mais aussi de l'administrateur système et réseau de l'environnement dans lequel se trouve la machine qui héberge le SGBD. Le système et les applications présentes sur la machine doivent être maintenus et mis à jour régulièrement pour éviter les problèmes. L'administrateur système doit également tenir à jour les comptes utilisateurs qui possèdent les droits d'accès à la machine et vérifier la qualité des mots de passe associés. Classiquement, une intrusion se fait en utilisant un compte « oublié », créé par exemple pour un besoin ponctuel, dont le mot de passe est assez simple pour être deviné. Le SGBD lui-même est une application parmi les autres ; il peut présenter des failles et doit faire partie du programme des mises à jour critiques. Cette tâche est généralement effectuée par l'administrateur du SGBD, qui doit également tenir à jour les comptes d'utilisateurs spécifiques du SGBD disposant de droits d'accès à la machine.

En résumé, le système sur lequel est hébergée la base de données doit faire l'objet d'un suivi permanent à tous les niveaux. Il n'est pas raisonnable d'installer un SGBD sur une machine dont le système ne peut être mis à jour ou dont les mises à jour sont disponibles trop tardivement. Toute application présente sur la machine peut recéler des failles et être le maillon faible de l'ensemble. Les informations de vulnérabilité et les correctifs sont fournis par les éditeurs de ces applications. Dans notre cas, on doit porter une attention particulière à une application importante : le SGBD. La réactivité des éditeurs constitue un critère décisif pour choisir un système et un SGBD. D'un point de vue humain, la qualité de l'administration du système et du réseau sur lequel se trouve la machine peut être également un élément du choix.

1.3 Pérennité du système

Une base de données se conçoit généralement pour de nombreuses années d'utilisation. Il est donc essentiel, avant de choisir une machine, son système d'exploitation ainsi que le logiciel de SGBD, de prendre en considération leurs pérennités respectives. Le choix peut alors se porter sur des ensembles moins performants mais que l'on considère comme plus viables en termes de durée de vie.

Voici quelques points à considérer :

- **Le suivi de la machine elle-même.** Par exemple, la durée de la garantie matérielle, la possibilité de retrouver les pièces etc.
- **La durée de vie du système d'exploitation.** Un système qui n'est plus mis à jour par son éditeur et qui présente des trous importants de sécurité se révèle un très mauvais choix. La plupart des éditeurs continuent à assurer une maintenance minimale pour les anciens systèmes, mais ce n'est pas systématique.
- **L'évolutivité du SGBD.** Un SGBD propriétaire qui n'offre pas les échanges de données avec d'autres logiciels handicape la migration vers un autre SGBD. De même que pour les points précédents, un SGBD qui présente des failles importantes de sécurité d'accès ou de fonctionnement et qui n'est plus mis à jour par l'éditeur constitue un problème potentiel qu'aucune autre précaution ne saurait combler.

Cette projection dans l'avenir n'est pas toujours évidente compte tenu du nombre de paramètres à prendre en compte. Il est difficile de prédire qu'un constructeur ou un éditeur ne

disparaîtra pas dans les années à venir. Il est important d'envisager dès la conception l'éventualité d'une migration vers un autre SGBD, éventuellement hébergé par un autre système.

1.4 Sauvegarde, redondance et accessibilité des données

Sauvegarde des données

Des **copies de sauvegarde** des données contenues dans la base, mais aussi des métadonnées (dictionnaire de données) du SGBD, doivent être réalisées régulièrement. Outre les problèmes de malveillance, un incendie ou une inondation peuvent réduire à néant des années de travail. Dans la mesure du possible, les sauvegardes ne doivent pas être stockées dans le même endroit que la machine. L'idéal étant de disposer d'une armoire de stockage adaptée qui soit ignifugée et située dans un endroit à l'abri des inondations. Si ce stockage se trouve dans un lieu différent, c'est encore mieux.

Compte tenu du volume des données utilisé actuellement, la sauvegarde peut poser des problèmes de taille de support physique de stockage. À cet effet, on utilise fréquemment des robots de sauvegarde spécialisés, manipulant des bandes que l'on peut regrouper pour former un espace de stockage suffisant. Les disques regroupés dans un dispositif autonome (type RAID) sont également très employés pour les sauvegardes en raison de leur prix relativement abordable. Ils apportent de plus une certaine sécurité liée à des mécanismes de redondance, qui leur assure un fonctionnement sans pertes de données en cas de panne de l'un des disques.

Redondance des données

Un procédé plus efficace et plus sûr consiste à disposer de copies de la base en plusieurs endroits géographiquement distincts. La plupart des SGBD sont capables d'effectuer des mises à jour sur des copies de la base de données situées sur d'autres machines *via* le réseau. La fréquence de ces mises à jour va dépendre de celle de modification des données. Cette méthode permet de plus d'améliorer la **disponibilité** des données :

- Si l'un des serveurs est inaccessible, un autre peut prendre le relais.
- Il est possible ainsi de répartir la charge sur plusieurs serveurs.

On obtient une **redondance** de machines et donc des données. Cela peut sembler paradoxal dans la mesure où une grande partie de l'ouvrage est consacrée à la suppression de la redondance, mais le concept mis en œuvre ici est différent. Les données sont simplement dupliquées pour se prémunir contre leur perte et augmenter leur disponibilité. Ce dispositif est complexe à gérer d'un point de vue du réseau et nécessite de disposer de plusieurs hébergements géographiquement distants (voir figure 6.1).

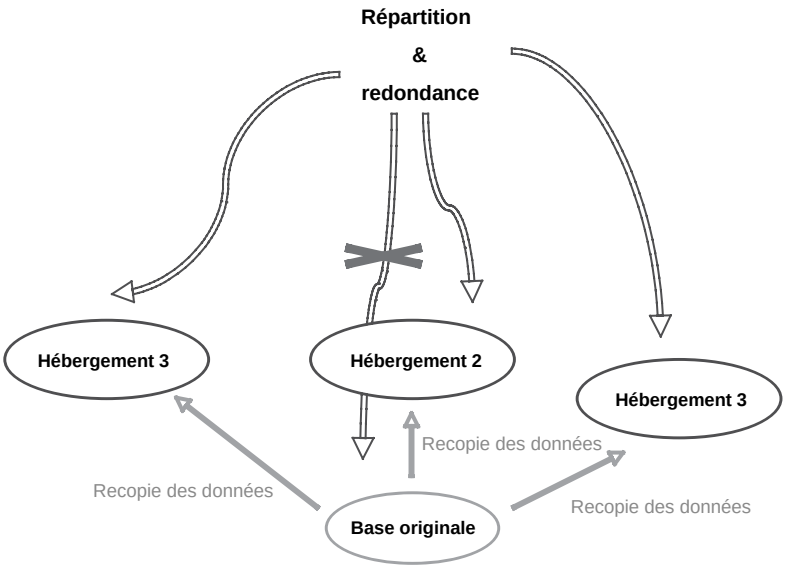


Figure 6.1

Redondance et répartition de charge.

2 Limitations d'accès au SGBD

Après avoir résolu les problèmes généraux d'accès à la machine, abordés dans la section précédente, il convient de restreindre l'accès au SGBD. Ces restrictions concernent les autorisations accordées ou non sur les différents éléments que contient le SGBD, mais aussi permettent de contrôler l'accès général au SGBD.

Ces opérations de gestion sont effectuées typiquement par l'administrateur de la base de données, qui définira des identifiants et leur affectera des droits, de manière totalement indépendante du système d'exploitation. Ces informations seront stockées dans le dictionnaire de données du SGBD. Il est possible d'accéder au SGBD, éventuellement à distance, sans posséder de compte sur la machine qui l'héberge. L'administrateur du SGBD doit donc gérer avec le plus grand soin ces identifiants, afin qu'ils deviennent pas un moyen de pénétrer dans le système d'exploitation en cas de faille dans le SGBD.

2.1 SQL et la gestion des utilisateurs

Les instructions qui permettent d'affecter les droits sont définies dans la norme SQL. Comme pour les autres éléments de la norme, les éditeurs ne les ont pas toujours intégrées totalement dans leurs produits. À l'inverse, ils ont souvent ajouté des instructions qui ne figurent pas dans la norme pour gérer ces droits. Classiquement, pour protéger une ressource, on commence par en interdire l'accès à tous. Puis on décrit ceux qui sont autorisés à l'utiliser (utilisateurs). On associe ensuite à chaque utilisateur un identifiant (mot de passe) pour éviter les emprunts d'identité.

Dans le principe, SQL ne considère pas réellement la notion d'utilisateur pour le SGBD tel qu'on le conçoit au sens d'un système d'exploitation. On peut envisager l'utilisation de plusieurs identifiants associés au même nom : cela correspondrait grossièrement à la possibilité d'utiliser des mots de passe spécifiques selon la ressource. Il s'agit donc d'une logique différente de celle qui consiste à associer strictement un mot de passe à un « compte » et à lui procurer des droits sur les ressources. La norme SQL considère que c'est l'objet à

protéger qui est au centre du processus : les droits d'utilisation de cet objet sont ensuite affectés à des couples de types (nom, identifiant).

Le SGBD stocke les informations suivantes pour chaque objet qu'il contient :

- **Le type de droit.** Sélection, création, destruction…
- **L'objet sur lequel s'appliquent les droits.** Une base de données, une table, une vue…
- **Le nom de l'utilisateur.**
- **L'identifiant, au sens du mot de passe.**
- **Le donneur des droits.**

Remarque

En pratique, la plupart des SGBD intègrent une instruction pour créer un utilisateur et lui associer un identifiant qui reste unique. Même si elle n'est pas définie réellement dans la norme SQL, elle est fréquemment de la forme :

```
CREATE USER <type de droit> IDENTIFIED BY <identifiant> ;
```

Cette instruction met à jour le dictionnaire de données du SGBD. Ce dernier utilise les informations de l'utilisateur pour authentifier la connexion au SGBD. Là encore, le mode de connexion dépend du SGBD utilisé. Par extension, celui-ci fournira en général une instruction du type DROP USER pour détruire l'utilisateur créé par la précédente commande.
Comme les informations du dictionnaire de données sont le plus souvent stockées dans des bases de données, on peut également les manipuler avec des instructions SQL classiques de types INSERT et DELETE. La spécificité de l'instruction CREATE USER est de posséder une instruction de cryptage du mot de passe.

Compte tenu de la remarque précédente, les éditeurs intègrent souvent la notion plus classique d'utilisateur dans un SGBD. Ce dernier permet en général de distribuer des autorisations à un ensemble d'utilisateurs qui constituent ainsi un **groupe**. Les tâches de gestion en sont facilitées, mais les groupes, qui représentent l'organisation de l'entreprise, sont parfois complexes à gérer. L'affectation de droits à une hiérarchie de groupes et sous-groupes peut relever du casse-tête.

Afin de résoudre ce problème, on dispose d'un autre modèle de distribution des droits : le **rôle**. Celui-ci désigne un assortiment de droits que l'on désire affecter à un objet du SGBD. C'est une vue inversée où les utilisateurs prennent le(s) rôle(s) dont ils ont besoin pour pouvoir travailler sur les objets du SGBD qui les concernent. L'instruction pour créer un rôle est la suivante :

```
CREATE ROLE consultation_seulement ;
```

L'affectation des droits à un rôle et la distribution de rôles à des utilisateurs sont présentées à la section suivante. La gestion des rôles, même s'ils font partie de la norme SQL, n'est pas proposée par tous les SGBD. Il en est de même pour les groupes d'utilisateurs ou tout simplement de la notion d'utilisateur qui n'existe pas toujours dans le SGBD.

En résumé, il n'y a pas de règle générale de gestion des autorisations de connexion au SGBD. L'initiative en est laissée à l'éditeur du SGBD. Cet aspect peut provoquer des soucis lors de la migration vers un autre SGBD.

2.2 SQL ET LA GESTION DES DROITS

Une fois que l'utilisateur est connecté au système, il convient de gérer son accès aux données. Une pratique courante pour les SGBD est que tout nouvel objet créé – une base de données par exemple – est par défaut inaccessible à tous. Les droits sont distribués ensuite à des utilisateurs, des groupes ou des rôles. Il existe ensuite une hiérarchie entre les différents objets du SGBD : une table est comprise dans une base de données ; un champ est compris dans une table, etc. Afin de simplifier la gestion, la plupart des SGBD permettent l'affectation de droits à un objet associé à tous les sous-objets qui constituent une branche de la hiérarchie.

Droits sur les objets du SGBD

Avec SQL, il est possible de spécifier des droits essentiellement sur les opérations de manipulation de tables (ou de vues considérées comme des tables) suivantes :

- interrogation (SELECT) ;
- insertion (INSERT) ;
- mise à jour (UPDATE) ;
- destruction (DELETE) ;
- référence à une table (REFERENCE).

Les bénéficiaires des droits sont les utilisateurs, les groupes ou les rôles. Les droits sont accordés par l'administrateur du SGBD. Cependant, il est possible de transférer la faculté de redistribuer ces droits à un autre utilisateur avec l'option WITH GRANT OPTION. L'instruction GRANT permet de réaliser l'association des droits et du bénéficiaire. Sa forme générale est la suivante :

```
GRANT <type de droit>
ON <objet>
TO <nom utilisateur>
```

Voici un exemple d'utilisation de GRANT.

L'administrateur crée un utilisateur « pastorius » avec l'identifiant « fender ».

```
CREATE USER pastorius IDENTIFIED BY fender ;
```

Il lui donne tous les droits sur la base de données 'pastorius_base' qu'il vient de créer. L'option 'ALL PRIVILEGES' sert à transmettre l'ensemble des droits dont dispose le donneur : en l'occurrence, l'administrateur dispose de tous les droits sur tous les objets. Le fait de préciser 'pastorius_base.*' signifie tous les sous-objets présents et à venir contenus dans la base de données 'pastorius_base'

```
CREATE DATABASE pastorius_base;
GRANT ALL PRIVILEGES ON pastorius_base.* TO 'pastorius' ;
```

L'utilisateur 'pastorius' crée la table 'jaco' dans la base de données 'pastorius_base' et donne l'accès de type « SELECT » à l'utilisateur 'nhop' préalablement créé.

```
USE pastorius_base;
CREATE TABLE jaco (NumMorceau INT PRIMARY KEY, Titre CHAR(50) ) ;
GRANT SELECT ON jaco TO 'nhop' ;
```

On obtient un message d'erreur : l'utilisateur 'pastorius' n'a pas le droit de retransmettre ses droits à un autre utilisateur. L'administrateur aurait dû entrer la commande suivante :

```
GRANT ALL PRIVILEGES ON pastorius_base.* TO pastorius WITH GRANT OPTION ;
```

Cette fois, la commande de transmission de droits peut être faite par l'utilisateur 'pastorius'

```
GRANT SELECT ON jaco TO 'nhop' ;
```

L'utilisateur 'nhop' a le droit d'effectuer toutes les opérations de consultation de la table 'jaco' de la base de données 'pastorius_base', il peut par exemple exécuter la séquence suivante :

```
USE pastorius_base;
SELECT * FROM jaco;
```

Lorsque l'on veut donner un droit à tous les utilisateurs du SGBD, on utilise le mot clé PUBLIC comme nom d'utilisateur.

```
GRANT SELECT ON jaco TO PUBLIC;
```

L'ensemble des utilisateurs peut alors interroger la table 'jaco' de la base de donnée 'pastorius_base' .On peut affiner ces permissions en ne les accordant que pour certains champs de la table. On spécifie alors pour le type de droit le ou les champs sur lesquels ils s'appliquent.

```
GRANT UPDATE Adresse ON jaco TO 'nhop' ;
```

L'utilisateur 'nhop' a le droit de mettre à jour le champ 'Adresse' de la table 'jaco'. Ce droit peut lui avoir été accordé par l'utilisateur 'pastorius' ou par l'administrateur du système. Si l'on ne spécifie pas le champ comme on l'a fait précédemment, le SGBD considère que le droit concerne tous les champs de la table.

Droits associés aux rôles

L'instruction GRANT permet également de donner un rôle à un utilisateur ou à un ensemble d'utilisateurs et s'utilise alors de la manière suivante :

```
GRANT <rôle>
 TO <liste des noms d'utilisateur séparés par des virgules>
```

Voici un exemple d'utilisation de GRANT avec les rôles. On considère les rôles suivants sur la table 'jaco' :

- consultation : interrogation ;
- utilisation : mise à jour et insertion ;
- gestion : destruction.

On crée les rôles et on met à jour les droits nécessaires.

```
CREATE ROLE consultation;
CREATE ROLE utilisation;
CREATE ROLE gestion;
GRANT SELECT ON jaco TO consultation;
GRANT UPDATE, INSERT ON jaco TO utilisation;
GRANT DELETE ON jaco TO gestion;
```

On affecte les rôles aux utilisateurs.

```
GRANT consultation TO 'pastorius', 'nhop', 'miller' ;
GRANT utilisation TO 'pastorius', 'miller' ;
GRANT gestion TO 'pastorius';
```

Les utilisateurs qui ont plusieurs rôles disposent des droits de tous les rôles auxquels ils appartiennent. Ainsi, l'utilisateur 'miller' possède les droits suivants sur la table : « INSERT, UPDATE, SELECT ». Pour retirer les droits accordés par l'instruction GRANT, on utilise l'instruction REVOKE**.** Elle est de la forme suivante :

```
REVOKE <type de droit>
ON <objet>
FROM <nom utilisateur>
```

Pour retirer les droits de l'utilisateur 'nhop' sur la table 'jaco', on procède comme suit :

```
REVOKE SELECT
ON jaco
FROM 'nhop' ;
```

Remarque

Seul l'utilisateur qui lui a donné ces droits peut les lui retirer. Dans notre cas, la commande précédente doit être lancée par l'utilisateur 'pastorius'.

Les droits donnés par plusieurs utilisateurs sont cumulatifs. Cela signifie que tous les droits accordés doivent être retirés pour qu'un utilisateur ne puisse plus effectuer les opérations. Ce « graphe » de permissions n'est pas toujours simple à gérer et devient rapidement de taille exponentielle. Par conséquent, les administrateurs des SGBD ont tendance à accorder parcimonieusement les possibilités de redistribuer des droits.

Pour aider l'administrateur, un bon SGBD procure une option de l'instruction de destruction d'un utilisateur, capable de détruire également tous les objets qu'il a créés. L'instruction DROP USER, non normalisée par SQL comme on l'a vu précédemment, s'utilise avec l'option CASCADE pour détruire les tables, les vues et autres que l'utilisateur a créés.

2.3 Utilisation des vues

La gestion des droits dans le SGBD, présentés à la section précédente, peut constituer un véritable casse-tête lorsque le nombre d'utilisateurs et d'objets augmente. En effet, les droits sur les tables et surtout sur les champs peuvent être définis par l'instruction GRANT, mais il devient vite fastidieux et particulièrement difficile de les modifier. Un des aspects de cette difficulté est le manque de lisibilité de l'ensemble des permissions.

La notion de rôle permet de « factoriser », ou rassembler, un ensemble de droits sur un ou plusieurs objets. Une approche complémentaire consiste à définir plus finement les objets sur lesquels on affecte les droits en se servant simplement des **vues SQL**. Celles-ci permettent de « cacher » certaines données à l'aide de critères de sélection précis, ce qui est impossible à réaliser d'une autre manière. De plus, les vues permettent de masquer aux utilisateurs la complexité de la structure des données, leur fournissant ainsi une interface plus commode avec la base de données.

Le mécanisme de création des vues a été abordé rapidement au chapitre 4. Les données ne sont pas stockées et sont recalculées à chaque utilisation de la vue. Une vue est le résultat d'une instruction SQL SELECT… : les possibilités de définition sont donc très étendues puisqu'il s'agit du cœur même du fonctionnement de SQL. Enfin, il est également possible de mettre à jour les données contenues dans les vues sous certaines conditions. Les permissions sur les vues sont ensuite accordées par les instructions de type GRANT utilisées précédemment. Voici quelques exemples de vues qui recourent à la base de données exemple 'casse'.

Vue utilisant une jointure simple

Le service marketing veut vérifier s'il n'y a pas de corrélations entre la couleur des véhicules vendus et la ville dans laquelle résident les clients qui les achètent. Il est inutile de fournir à ce service la procédure pour obtenir ces informations qui nécessite de lier les trois tables 'voiture', 'vente' et 'client'. On crée une vue qui contient uniquement la projection de ces deux informations et qui sélectionne les voitures vendues.

```
CREATE VIEW couleur_ville (Couleur, Ville)
AS SELECT voiture.couleur, client.ville
FROM voiture JOIN vente JOIN client ON voiture.NumVoit=vente.Numvoit
AND client.NumAch = vente.Numach ;
```

On donne les droits de visualiser ces informations aux personnes du service marketing qui sont les utilisateurs 'nhop' et 'miller'.

```
GRANT SELECT ON couleur_ville TO 'nhop', 'miller' ;
```

Vue utilisant une jointure plus élaborée

On souhaite que tous les utilisateurs puissent consulter le catalogue des voitures non encore vendues. Cette liste de voitures est accessible par une requête plus complexe que la précédente, de type jointure externe. Il s'agit d'un cas d'utilisation d'une vue, très pratique pour les utilisateurs qui n'ont pas à connaître les notions avancées de jointure externe.

```
CREATE VIEW catalogue (NumVoit, Marque, Type, Couleur)
AS SELECT voiture.NumVoit, voiture.Marque, voiture.Type, voiture.Couleur
FROM voiture LEFT OUTER JOIN vente ON voiture.NumVoit=vente.Numvoit
WHERE vente.Numvoit IS NULL;
```

On donne la permission à l'utilisateur PUBLIC, c'est-à-dire à tous les utilisateurs.

```
GRANT SELECT ON catalogue TO PUBLIC;
```

Vue avec possibilité de mise à jour

Le service comptabilité doit pouvoir accéder aux informations de vente et de clientèle afin de facturer et calculer le chiffre d'affaires. En outre, ce service a besoin de mettre à jour le prix de vente.

```
CREATE VIEW journal_compta (Date_de_vente, Prix_de_vente, Nom_client, Ville_client)
AS SELECT vente.DateVent, vente.Prix, client.Nom, client.Ville
FROM vente JOIN client ON vente.NumAch=client.NumAch
WITH CHECK OPTION ;
```

L'option CHECK OPTION permet les mises à jour des données au travers de la vue.

On donne les droits à un rôle que les utilisateurs 'pastorius' et 'miller' vont utiliser.

```
CREATE ROLE comptabilite ;
GRANT SELECT ON journal_compta TO comptabilite;
GRANT UPDATE Prix_de_vente ON journal_compta TO comptabilite;
```

On donne le rôle 'comptabilité' aux utilisateurs 'pastorius' et 'miller'.

```
GRANT comptabilite TO 'pastorius', 'miller'
```

L'utilisation des vues permet de définir des objets dynamiques puisqu'une vue est le résultat d'une requête et n'est jamais stockée. Les vues s'utilisent en complément du système général de droits présenté à la section précédente. Il s'agit de la bonne solution pour éviter la complexité de description des permissions sur la totalité des champs. Comme pour le reste de la norme SQL, tous les SGBD ne proposent pas les vues.

3 Transactions

Cette section présente plusieurs outils procurés par les SGBD pour protéger les opérations effectuées sur les données. Les incidents liés aux données dans un SGBD proviennent essentiellement de l'**accès concurrent** à ces dernières par plusieurs utilisateurs. Ces problèmes sont habituellement résolus par les mécanismes associés aux **transactions** présentées dans cette section. Celles-ci permettent également de résoudre les pertes dues aux

erreurs de manipulation ou celles liées aux erreurs dans le traitement des données, par exemple en cas de panne matérielle.

3.1 Accès concurrent aux données

La section précédente aborde la définition des accès et des permissions sur les différents objets gérés par le SGBD. Implicitement, cela signifie que plusieurs utilisateurs ou applications peuvent accéder aux données en même temps. De surcroît, la plupart des machines sont connectées à un réseau, ce qui augmente encore le nombre potentiel d'utilisateurs simultanés.

Lecture(s) étrange(s)

L'accès multiple en lecture ne pose pas habituellement de problèmes, mais que se passe-t-il lorsqu'un ou plusieurs utilisateurs décident de modifier les mêmes données au même moment ? On considère la séquence d'instructions suivante appliquée à la base de données « casse ». On suppose que tous les utilisateurs disposent de tous les droits sur tous les objets (tables et champs) de cette base de données :

- L'utilisateur 'pastorius' consulte la liste des voitures non vendues.
- L'utilisateur 'nhop' enregistre la vente d'une voiture.
- L'utilisateur 'pastorius' relance la même requête et trouve un résultat différent.
- L'utilisateur 'nhop' invalide la vente de cette voiture.
- L'utilisateur 'pastorius' pensant s'être trompé relance la même requête qui aboutit au résultat initial.

Pour trois exécutions de la même requête, l'utilisateur 'pastorius' va obtenir des résultats différents. On peut considérer la même séquence exécutée dans un ordre différent.

- L'utilisateur 'pastorius' consulte la liste des voitures non vendues.
- L'utilisateur 'pastorius' relance la requête et trouve le même résultat.
- L'utilisateur 'nhop' enregistre la vente d'une voiture.
- L'utilisateur 'nhop' invalide la vente de cette voiture.
- L'utilisateur 'pastorius' relance la requête précédente et retrouve le même résultat.

Comment décider de l'ordre dans lequel exécuter les instructions ? En pratique, le SGBD ne dispose pas d'éléments de comparaison qui lui permettraient d'ordonner la série d'instructions afin que cela passe inaperçu comme dans la deuxième série. On pourrait utiliser ici le terme de **lecture fantôme** pour qualifier les problèmes rencontrés : une donnée apparaît puis disparaît.

Incohérence de résultats

On peut imaginer une autre série d'instructions qui réalisent des modifications de données effectuées par différents utilisateurs. En raison de l'augmentation des frais de structure, le service comptable a décidé d'une augmentation générale du prix de vente de 5 %. Afin que cette dernière passe inaperçue et dans le cadre d'une campagne de communication, le service marketing offre 100 euros de ristourne sur tout le catalogue pendant un mois. L'utilisateur 'pastorius' appartient au service comptable et l'utilisateur 'nhop' au service marketing. Les vérifications sont effectuées par l'utilisateur 'miller' de la direction (voir figure 6.2).

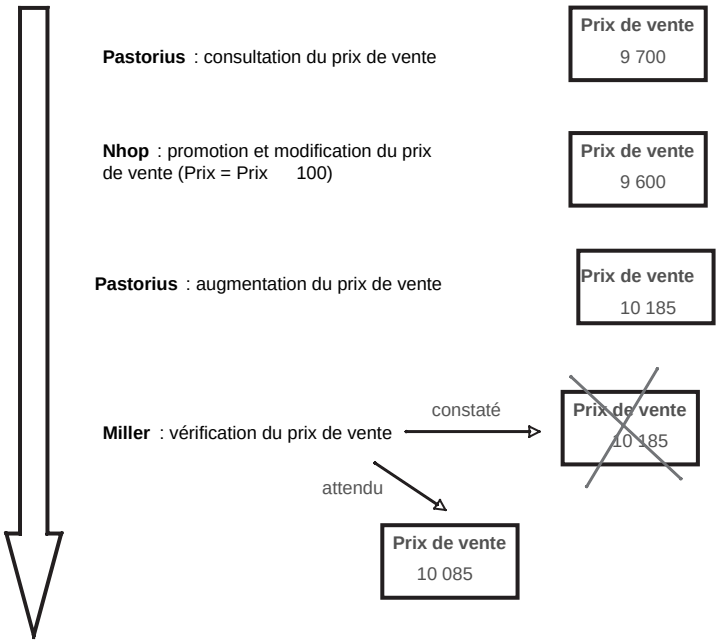
- L'utilisateur 'pastorius' consulte le prix de vente de la voiture 'X'.

- L'utilisateur 'nhop' effectue la modification de promotion sans vérifier le prix de vente (Prix_vente = Prix_vente – 100).
- L'utilisateur 'pastorius' effectue la modification d'augmentation (Prix_vente = Prix_vente × 1,05).

L'utilisateur 'miller' vérifie le résultat de l'opération de modification du prix de vente et constate une différence avec le résultat attendu : le prix est égal à (Prix_vente – 100) × 1,05, alors qu'il s'attendait à un prix égal à (Prix_vente × 1,05) – 100.

Figure 6.2

Déroulement de la séquence d'instructions de mise à jour du prix de vente.



On obtient alors une « incohérence » due à la séquence de modification des données. Les opérations effectuées sur le champ 'Prix' ne sont en effet pas commutatives. Là encore, le SGBD ne dispose pas d'éléments lui permettant d'ordonner correctement les opérations. En revanche, pour certains prix de vente, la mise à jour pourra être correcte.

Il existe bien d'autres types d'incohérences qui peuvent être provoquées par l'accès concurrent aux données. On peut citer le cas des **anomalies de lecture** : « Deux lectures successives ne donnent pas le même résultat. » Cela survient lorsque la modification d'une donnée est effectuée par un processus concurrent entre deux lectures successives. Le SGBD traite naturellement les requêtes de manière séquentielle, dans l'ordre de leur arrivée. Si l'on augmente le nombre d'utilisateurs, ce type de problèmes peut évidemment se multiplier.

Verrous

On a donc besoin de disposer d'un mécanisme qui garantisse une certaine « exclusivité » sur les données lors des opérations de mise à jour. Un tel mécanisme existe depuis longtemps en informatique pour résoudre ce problème : il s'agit des **verrous**. L'idée est simple : on bloque une ressource pour effectuer les opérations et on la libère dès que les opérations sont effectuées. Cette méthode semble résoudre le problème ; elle présente cependant quelques pièges et doit être utilisée avec prudence. En effet, on peut rapidement parvenir à une situation de blocage que l'on nomme **étreinte fatale** (*deadlock* en anglais) ou parfois

interblocage. Pour illustrer cette situation, on suppose que deux vendeurs veulent mettre à jour la base de données 'casse' :

- L'utilisateur 'pastorius' veut insérer une vente de voiture dans la table 'vente' et constate à cette occasion une erreur sur la couleur dans la table 'voiture' qu'il veut modifier.
- L'utilisateur 'miller' veut insérer à la fois une nouvelle voiture dans la table 'voiture' et la mention de sa vente dans la table 'vente'.

La séquence d'instruction peut être la suivante :

- L'utilisateur 'pastorius' pose un verrou sur la table 'vente'.
- L'utilisateur 'miller' pose un verrou sur la table 'voiture'.
- L'utilisateur 'pastorius' a terminé la mise à jour de 'vente' et veut réaliser celle de 'voiture'… mais la table 'voiture' est verrouillée par 'miller'.
- L'utilisateur 'miller' a terminé la mise à jour de 'voiture' et veut procéder à celle de 'vente'… mais la table 'vente' est verrouillée par 'pastorius.

On parvient à une situation où les deux utilisateurs attendent que l'autre libère la ressource pour continuer. Évidemment, c'est une attente infinie qui se profile pour chacun. D'autres phénomènes de blocage peuvent survenir dans l'utilisation des verrous. On parle parfois dans ce cas de **famine**. Voici un exemple illustrant cette possibilité.

L'utilisation classique des verrous pour protéger l'accès à une ressource est la suivante (algorithme des « lecteurs-rédacteurs ») :

- Tous les lecteurs peuvent lire en même temps.
- Si un rédacteur veut écrire, aucun lecteur ne doit plus utiliser la donnée.
- Si un rédacteur est en train d'écrire, aucun lecteur ne peut lire la donnée et aucun autre rédacteur ne peut y accéder (accès exclusif).

S'il existe un grand nombre de lecteurs qui se succèdent en lecture sur la donnée, l'attente pour un rédacteur qui voudrait la modifier peut alors devenir quasi infinie.

Laisser la gestion des verrous à un utilisateur est donc délicat et peut conduire à des blocages du système. Les SGBD performants disposent d'outils capables de détecter et résoudre ces phénomènes, voire de les prévenir le cas échéant. Les algorithmes utilisés à cet effet dépassent le cadre de cet ouvrage.

3.2 Mécanisme des transactions

Pour résoudre les problèmes d'accès concurrents vus précédemment, les SGBD proposent aux utilisateurs un outil que l'on nomme une **transaction**. L'idée est de considérer un ensemble d'instructions comme une seule instruction. L'ensemble d'instructions est dit **unitaire** ou **atomique**.

Propriétés des transactions

La notion d'atomicité peut être décrite en ces termes :

- Soit le SGBD est capable d'exécuter toutes les instructions qui composent une transaction et il effectue les mises à jour provoquées par ces instructions.
- Soit il n'y parvient pas et il remet la base de données dans l'état cohérent précédent le début de l'exécution des instructions de la transaction.

Les propriétés que doivent vérifier les transactions sont résumées par le terme **ACID :**

- **Atomicité.** Une transaction est atomique : elle est exécutée entièrement ou abandonnée.
- **Cohérence.** La transaction doit se faire d'un état cohérent de la base vers un autre état cohérent.
- **Isolement.** Des transactions simultanées ne doivent pas interférer entre elles.
- **Durabilité.** La transaction a des effets permanents même en cas de panne.

Les qualités des transactions sont un argument de vente pour un SGBD. Il est en effet complexe de trouver un compromis entre la sécurité et les performances. Bien qu'il s'agisse d'un mécanisme éprouvé qui existe depuis fort longtemps, les transactions ainsi que les techniques de gestion des accès concurrentiels restent des sujets de recherches importants dans les laboratoires. Les transactions font partie de la norme SQL.

Remarque

Les transactions permettent de résoudre les problèmes liés aux accès concurrentiels, mais également de traiter le cas de la reprise en cas de panne. Le SGBD est ainsi capable de remettre la base de données dans l'état cohérent où elle se trouvait avant le début de la transaction qui a échoué pour cause de panne. Les transactions sont également employées pour annuler des erreurs de traitement éventuelles. En effet, grâce à ce mécanisme, on peut revenir à l'état initial dans lequel se trouvait la base de données avant le début de la série de mauvaise(s) manipulation(s) sur la base de données. Il s'agit certainement de l'utilisation principale des transactions.

Réalisation des transactions dans les SGBD

Pour mettre en œuvre les transactions, le SGBD doit offrir **l'exclusivité d'accès** aux données ainsi que la possibilité **d'annuler** des modifications effectuées. Afin de garantir l'accès exclusif aux données, le SGBD utilise les verrous vus précédemment. Ces mécanismes sont associés à des algorithmes de protection sophistiqués afin de prévenir les problèmes des méthodes de verrouillage.

La possibilité de revenir en arrière repose sur l'utilisation d'un journal de transactions. Pour ce faire, le SGBD garde une trace de toutes les requêtes de la transaction et des données qui sont modifiées par ces instructions. Ce mécanisme est identique à celui qui est utilisé pour les systèmes de fichiers modernes dans les systèmes d'exploitation. Puisque la plupart des écritures se font en mémoire pour gagner du temps, le système de fichiers n'est pas forcément cohérent en cas de panne de courant qui entraîne la perte de toutes les informations se trouvant en mémoire. On conserve donc un journal de l'ensemble des lectures et écritures afin de pouvoir reconstituer un système cohérent. En utilisant ce journal, à partir d'un état courant du système, on est capable de réexécuter les instructions qui n'ont pu l'être effectivement ou on revient en arrière si ce n'est pas possible.

La description faite ci-dessus est évidemment simpliste par rapport à la réalité plus complexe des SGBD. Les méthodes de verrouillage implémentées dans ces derniers sont bien plus élaborées que celles présentées ici. Il en est de même pour les mécanismes de journalisation qui disposent de plusieurs niveaux.

Différents niveaux de transaction

La norme SQL fixe plusieurs degrés de qualité pour les transactions : on parle de niveaux **d'isolation**. C'est-à-dire qu'une transaction pourra être rendue plus ou moins perméable aux autres transactions. Les niveaux standard sont les suivants :