CHAPITRE 1

Introduction aux bases de données

L'objectif de ce court chapitre est d'introduire la problématique des bases de données. Nous commencerons, section 1.1 par définir la notion de bases de données puis présenterons les principaux modèles de bases de données section 1.2. Nous nous intéresserons enfin, section 1.3, aux Systèmes de Gestion de Bases de Données (SGBD). Le premier TD, section 1.4, est particulier, car très informel. Son objectif est pourtant essentiel : sensibiliser le lecteur aux problèmes auxquels la discipline des bases de données apporte des solutions éprouvées.

Sommaire

1 Introduction aux bases de données

1.1 Qu'est-ce qu'une base de données

Il est difficile de donner une définition exacte de la notion de bases de données. Une définition très générale pourrait être :

Définition 1.1 – **Base de données** – *Un ensemble organisé d'informations avec un objectif commun.*

Peu importe le support utilisé pour rassembler et stocker les données (papier, fichiers…), dès lors que des données sont rassemblées et stockées d'une manière organisée dans un but spécifique, nous pouvons parler de base de données.

Plus précisément, nous appellerons base de données un ensemble structuré et organisé permettant le stockage de grandes quantités d'informations afin d'en faciliter l'exploitation (ajout, mise à jour, recherche de données). Bien entendu, dans le cadre de ce cours, nous nous intéressons aux bases de données informatisées.

Définition 1.2 – **Base de données informatisée** – *Une base de données informatisée est un ensemble structuré de données enregistrées sur des supports accessibles par l'ordinateur, représentant des informations du monde réel et pouvant être interrogées et mises à jour par une communauté d'utilisateurs.*

1.2 Problématique de la cohérence des données

La création d'une base de données répond au besoin de rassembler des données qui possèdent un lien entre elles, dans le but de retrouver de l'information en utilisant des critères de recherche basés sur le contenu de cette information. Prenons l'exemple d'une base de données sur des albums musicaux qui mémorise, pour chaque album, le genre, l'artiste et le titre de l'album. Un extrait d'une telle base est représenté figure 1.1.

Genre	Artiste	Titre
Alternative Rock	les Wampas	Les Wampas vous aiment
Altern Rock	Les Wampas	Tutti frutti
Alternative Rock	Wampas	Les bottes rouges
Altern Rock	Les Shériff	Les deux doigts dans la prise
Folk-Rock	Joan Osborne	Righteous Love
Folk/Rock	Leonard Cohen	Songs From A Room

FIGURE 1.1: Extrait d'une base de données qui représente, pour chaque album, le genre, l'artiste et le titre

La condition *sine qua non* pour garantir la faisabilité et la qualité d'une recherche de données par le contenu est la cohérence des données. Les données représentées sur la figure 1.1 souffrent de plusieurs problèmes qui compromettent leur consultation. Par exemple, le groupe *Les Wampas* est représenté de trois manières différentes dans la base :

les Wampas, *Les Wampas* et *Wampas*. Ainsi, une recherche sur la chaîne *Les Wampas* comme groupe ne permet de trouver que l'album *Tutti frutti* alors que le groupe possède trois albums dans la base. De la même manière, les genres *Alternative Rock* et *Folk-Rock* possèdent chacun deux représentations.

La cohérence des données est la problématique fondamentale des bases de données. La première et la plus importante des réponses à ce problème consiste à limiter au maximum la redondance d'informations. Nous verrons tout au long de cet ouvrage un nombre important de méthodes qui permettent d'assurer la cohérence des données. La plupart de ces méthodes s'appliquent au moment de la conception d'une base de données. L'étape de modélisation conceptuelle n'est donc pas à négliger et nous y consacrerons un chapitre entier (chapitre 2).

1.3 Système de gestion de bases de données

La cohérence des données peut également être mise à mal lors des phases d'insertion ou de modification des données, lors de pannes physiques du système ou encore lors d'accès concurrents à la base de données. La résolution de tous ces problèmes n'est heureusement pas à la charge des concepteurs d'applications qui utilisent des bases de données, mais est déléguée au *Système de Gestion de Bases de Données* (SGBD). Le SGBD est un ensemble de programmes qui assure la gestion et l'accès à une base de données. Nous y reviendrons dans la section 1.3. Un SGBD est caractérisé par le modèle de description des données qu'il supporte. Ce modèle peut être hiérarchique, réseau, relationnel ou bien objet. Ces différents modèles sont brièvement passés en revue dans la section qui suit (section 1.2).

Le modèle qui s'est naturellement imposé depuis plus de trente ans est le modèle relationnel (cf. chapitre 3), avec depuis quelques années une éventuelle extension objet. Les SGBD qui supportent ce modèle implémentent le langage SQL (cf. chapitre 4) qui constitue le langage d'accès normalisé aux bases de données relationnelles.

2 Modèles de bases de données

2.1 Historique

Les différents modèles de données élaborés pour structurer l'information dans une base de données reposent sur des principes théoriques empruntés au monde de la recherche en informatique.

Le premier modèle à s'attaquer au problème de la redondance des données fût développé par IBM, dans le cadre du programme spatial Apollo de la NASA, pendant les années 1960. Il s'agit du modèle hiérarchique décrit section 1.2.2.

À la fin des années 1960, Charles William Bachman proposa le modèle réseau, décrit section 1.2.3, qui généralise le modèle hiérarchique. Les premières spécifications de ce modèle furent publiées en 1969 par le groupe de travail DBTG (*Data Base Task Group*) de l'organisme américain CODASYL (*Conference On DAta SYstems Languages*). C.W. Bachman reçut le Prix Turing en 1973 pour ses contributions sur les technologies des bases de données.

Durant cette période, les ordinateurs évoluèrent rapidement en terme de puissance, de diffusion et de coût. Cette évolution permit aux modèles informatiques et aux langages de programmation d'atteindre un niveau d'abstraction suffisant pour les rendre indépendants d'une architecture spécifique d'ordinateur. C'est dans ce contexte favorable que Edgar Frank Codd, directeur de recherche du centre IBM de San José, publia en 1970 un article (Codd, 1970) où il proposait de stocker des données hétérogènes dans des tables. Ce modèle, qualifié de relationnel, était à l'époque considéré comme une curiosité intellectuelle. En effet, il n'était pas évident que les tables puissent un jour être gérées de manière efficace par un ordinateur. Ce scepticisme n'a cependant pas empêché Codd de poursuivre ses recherches et le modèle relationnel, décrit section 1.2.4, s'imposa rapidement.

Le paradigme orienté objet en programmation est né avec le langage Simula (*Simple universal language*) en 1967. Il est donc antérieur au modèle relationnel. Mais il fallut attendre les années 1980 pour observer le plein essor de la programmation orientée objet, et pratiquement les années 1990 pour son arrivée dans le monde industriel, avec des langages comme C++, Java ou Python. La notion de base de données objet (cf. section 1.2.5) s'est également précisée au début des années 1990, avec notamment la publication du manifeste *The Object-Oriented Database System Manifesto* d'Atkinson *et al.*. (1989) à la conférence DOOD (*Conference on Deductive and Object-Oriented Databases*). Cependant, le développement des SGBD objet s'est rapidement heurté à la nécessité de compatibilité avec l'existant, c'est-à-dire avec le modèle relationnel. D'où l'émergence du modèle relationnel-objet dont l'objectif est d'étendre le modèle relationnel avec les concepts essentiels de l'objet. Cette extension objet du modèle relationnel a fait l'objet d'une nouvelle norme en 1999 : la norme SQL-3.

2.2 Modèle hiérarchique

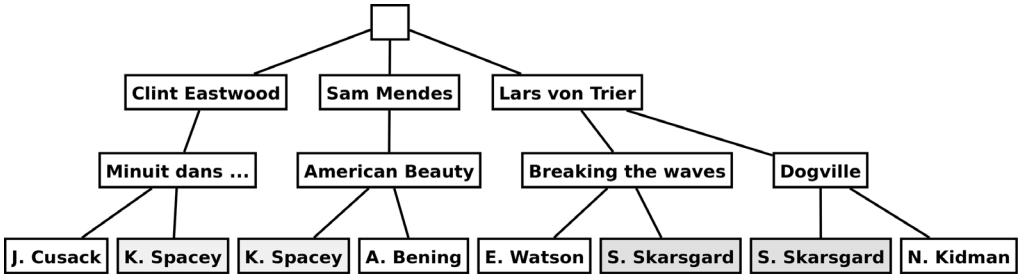

FIGURE 1.2: Modèle hiérarchique

Les modèles conceptuels de cette époque sont encore très liés à l'organisation des fichiers sur un ordinateur. Ainsi, un peu comme dans un système de fichiers, un SGBD hiérarchique lie des enregistrements dans une structure arborescente de façon à ce que chaque enregistrement n'ait qu'un seul possesseur. Une base de données hiérarchique peut donc être considérée comme un arbre comme celui illustré par la figure 1.2. Le modèle

hiérarchique a tout d'abord été conçu et utilisé pour le programme Apollo de la NASA. Il est aujourd'hui encore utilisé dans le monde de l'entreprise par le biais d'anciens systèmes comme IMS (*Information Management System*) d'IBM ou comme TOTAL de Cincom.

Les liens hiérarchiques entre les différents types de données peuvent rendre très simple la réponse à certaines questions, mais plus difficile la réponse à d'autres formes de questions. Si le principe de relation *1 vers N* n'est pas respecté (par exemple, sur l'illustration de la figure 1.2, Kevin Spacey et Stellan Skarsgard ont tous les deux joué dans deux films), alors le modèle hiérarchique n'est plus adapté, d'où l'introduction du modèle réseau.

2.3 Modèle réseau

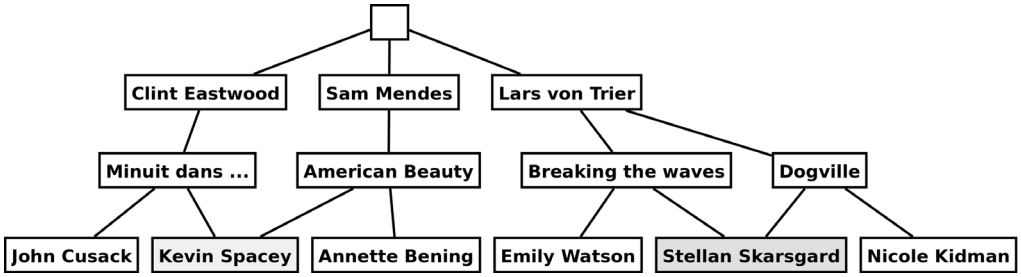

FIGURE 1.3: Modèle réseau

Le modèle réseau est une extension du modèle hiérarchique qui permet de lever de nombreuses difficultés inhérentes à ce dernier grâce à la possibilité d'établir des associations qui peuvent conduire aussi bien à des arbres qu'à des réseaux. Par exemple, l'illustration de la figure 1.3 montre maintenant que le fait que Kevin Spacey et Stellan Skarsgard ont tous les deux joué dans deux films peut être représenté sans avoir besoin de les faire figurer chacun deux fois. Parmi les SGBD réseaux encore utilisés dans les entreprises, nous pouvons citer IDMS de Computer Associates, IDS2 de Bull ou encore EDMS de Xerox.

De la même manière que dans un modèle hiérarchique, pour retrouver une donnée dans un modèle réseau, il faut connaître le chemin d'accès (les liens), ce qui rend les programmes dépendant de la structure de données. Pour cette raison, le modèle hiérarchique et le modèle réseau sont aujourd'hui inadaptés en tant que modèles de conception.

2.4 Modèle relationnel

Chercheur au centre de recherche d'IBM San-José à la fin des années 1960, Edgar Frank Codd n'était pas satisfait par les modèles de l'époque. Il chercha un nouveau modèle, plus simple, pour gérer de grandes quantités de données. Selon lui, ce modèle devait garantir un haut degré d'indépendance avec la représentation interne des données et fournir une base solide pour traiter le problème de la cohérence et de la redondance des données. Mathématicien de formation, il était persuadé de pouvoir s'appuyer sur la

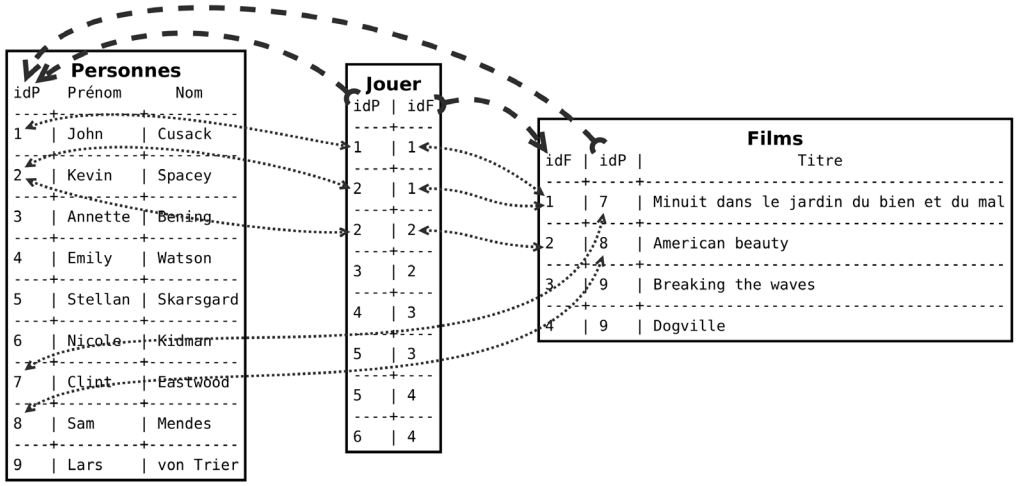

FIGURE 1.4: Modèle relationnel

théorie des ensembles et la logique des prédicats (ou logique du premier ordre). En 1970, Codd (1970) publia un article où il proposait de stocker des données hétérogènes dans des tables comme l'illustre la figure 1.4. Un premier prototype de Système de gestion de bases de données relationnelles (SGBDR) fut développé dans les laboratoires d'IBM. Depuis les années 1980, cette technologie a mûri et a été largement adoptée par le monde des entreprises.

Tout en développant le modèle relationnel, Codd mit au point un langage de manipulation des données non procédural où l'utilisateur définit le résultat qu'il attend plutôt que la manière de l'obtenir. Il développa ainsi le langage SEQUEL (*Structured English Query Language* en anglais), rebaptisé par la suite SQL (*Structured Query Language*). Le langage SQL fut normalisé dès 1986 par l'ANSI (*American National Standard Institute*) et adopté comme norme internationale par l'ISO (*International Organization for Standardization*) en 1987.

L'ensemble de cet ouvrage se situe dans le cadre de ce modèle.

2.5 Modèles orientés objet

✓ Modèle objet

Une base de données est généralement l'un des composants d'une application. Il est donc important que base de données, langage de programmation et méthodologie de développement de l'application forment un tout cohérent. Les bases de données orientées objet s'intègrent donc mieux et plus naturellement dans la mouvance de la conception orientée objet et des langages de programmation objet comme C++, Java ou Python. L'objectif principal des systèmes de gestion de bases de données objet (SGBDO) est ainsi d'unifier deux technologies : les SGBD et la programmation orientée objet (POO).

La notion de bases de données objet s'est précisée au début des années 1990 avec, notamment, la publication du manifeste de Atkinson *et al.*. (1989). L'une des difficultés majeures du modèle objet provenait de l'absence de standard. Aussi, à l'initiative de Sun Microsystems, un groupe de travail fut créé en 1991 : ODMG (*Object Data Management Group*). Ce groupe rassemblait cinq constructeurs de SGBDO : O2 Technology, Objectivity, Object Design, Ontos et Versant. L'ODMG a proposé, entre autres, une extension objet du langage SQL appelée OQL. Mais le développement du modèle objet se heurtait à la nécessité de compatibilité avec l'existant, c'est-à-dire avec le modèle relationnel. D'où l'émergence du modèle relationnel objet. L'ODMG disparut en 2001 supplanté par JDO (*Java Data Objects*) de Sun Microsystems.

✓ Modèle relationnel-objet

Le modèle relationnel-objet a pour objectif d'étendre le modèle relationnel pour lui conférer un certain nombre de caractéristiques essentielles du modèle objet. Ainsi, la totalité des fonctions d'un SGBDR classique est préservée puisqu'un SGBDR Objet (SGBDRO) est entièrement compatible avec le modèle relationnel classique. Cette extension objet du modèle relationnel fit l'objet d'une nouvelle norme en 1999 : la norme SQL-3. Le SQL-3 étend le langage de requêtes SQL en lui apportant notamment les appels d'opération, la surcharge des opérateurs, le support de l'héritage… D'une certaine manière, SQL3 entre en concurrence avec OQL de l'ODMG. Mais l'avantage de SQL-3 est d'être adopté par tous les grands constructeurs, comme IBM, Oracle et Sybase, et d'être impliqué dans un processus de standardisation international.

✓ Mapping objet-relationnel

La tendance actuelle est de faire cohabiter un langage de programmation orientée objet et une base de données relationnelle. La solution apportée pour faire cohabiter ces deux mondes est le *mapping objet-relationnel*. Cette technique permet de fournir un service de persistance au monde objet. Cependant, seul un expert maîtrisant simultanément les concepts objet et relationnels peut implémenter un tel mapping. Ce surcoût est évalué, en moyenne, à 30% du coût d'un projet Java.

Pour résoudre ce problème, des *framework*[1] de mapping objet-relationnel, également appelés framework de *persistance objet des données*, commencent à voir le jour. Dans cette optique, Sun Microsystems propose, pour le langage Java, une interface puissante : JDO (*Java Data Objects*). L'interface JDO définit en effet un concept de persistance transparent pour les objets Java en s'appuyant sur n'importe quel support de persistance (SGBDR, SGBDO, WebServices…). Toujours pour le langage Java, le framework actuellement le plus populaire est probablement Hibernate. Il s'agit d'un framework ouvert (*open source*) qui s'interface avec une base de données relationnelle pour remplacer les accès à la base de données par des appels à des méthodes objet de haut niveau.

1. Un *framework* est un ensemble de bibliothèques, d'outils et de cadres méthodologiques qui permettent d'assister, d'accélérer et de fiabiliser le développement d'applications complexes et de faciliter leur maintenance.

3 Système de gestion de bases de données (SGBD)

La gestion et l'accès à une base de données sont assurés par un ensemble de programmes qui constituent le *Système de Gestion de Bases de Données* (SGBD). Un SGBD héberge généralement plusieurs bases de données, qui sont destinées à des logiciels ou des thématiques différents. Un SGBD doit permettre l'ajout, la modification et la recherche de données. Quel que soit le modèle implémenté (réseau, relationnel, objet…), les principaux objectifs d'un SGBD sont de masquer la représentation physique des données et d'assurer la protection et la cohérence de ces données dans le cadre d'un environnement multi-utilisateur impliquant des accès concurrents.

3.1 Objectifs des SGBD

Nous énumérons et détaillons dans cette section les principaux objectifs que doivent remplir les SGBD.

Indépendance physique des données : La représentation interne des données et les méthodes d'accès au système de fichier doivent être transparentes pour l'utilisateur.

Indépendance logique des données : Un même ensemble de données peut être vu différemment par des utilisateurs différents. Toutes ces visions personnelles des données sont définies dans le *schéma externe* et doivent être intégrées dans une vision globale appelée *schéma conceptuel*. Une réorganisation, même profonde, du *schéma conceptuel* ne doit pas avoir d'impact sur le *schéma externe*.

Accès aux données : L'accès aux données se fait par l'intermédiaire d'un Langage de Manipulation de Données (DML). Il est crucial que ce langage permette d'obtenir des réponses aux requêtes en un temps raisonnable. Le DML doit donc être optimisé, minimiser le nombre d'accès disques, et tout cela de façon totalement transparente pour l'utilisateur.

Administration centralisée des données : Toutes les données doivent être centralisées dans un réservoir unique commun à toutes les applications. En effet, des visions différentes des données (entre autres) se résolvent plus facilement si les données sont administrées de façon centralisée.

Non-redondance des données : Afin d'éviter les problèmes lors des mises à jour, chaque information ne doit être représentée qu'une seule fois dans la base.

Cohérence des données : Les données sont soumises à un certain nombre de contraintes d'intégrité qui définissent un état cohérent de la base. Elles doivent pouvoir être exprimées simplement et vérifiées automatiquement à chaque insertion, modification ou suppression de données. Les contraintes d'intégrité sont décrites dans le Langage de Description de Données (LDD).

Partage des données : Il s'agit de permettre à plusieurs utilisateurs d'accéder aux mêmes données au même moment de manière transparente. Si ce problème est simple à résoudre quand il s'agit uniquement d'interrogations, cela ne l'est plus quand il s'agit de modifications dans un contexte multi-utilisateur. En effet, il faut permettre à plusieurs utilisateurs de modifier la même donnée « en même