MP/MP*
PC/PC*
PSI/PSI*
PT/PT*

Serge Bays

PRÉPAS SCIENCES

COLLECTION DIRIGÉE PAR **BERTRAND HAUCHECORNE**

INFORMATIQUE
TRONC COMMUN

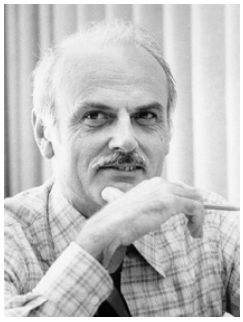
NOUVEAUX PROGRAMMES !

- Cours complet et détaillé
- Nombreux exemples
- Vrai/Faux
- Exercices avec indications
- Corrections et commentaires

ellipses

Chapitre 1
Organisation des données

UN SCIENTIFIQUE



Natif du sud de l'Angleterre, **Edgar Franck Codd** (1923-2003) suit des études de mathématiques et de chimie à Exeter puis à Oxford. Pendant la guerre, il sert comme pilote dans la Royal Air Force. Par la suite, il s'installe en Amérique du Nord (États-Unis et Canada) et se spécialise en informatique ; il s'intéresse plus particulièrement à la gestion des bases de données mais travaille aussi sur les automates cellulaires. Ses travaux furent récompensés par le prix Turing en 1981.

Un peu d'histoire

L'informatique s'est révélée d'une très grande efficacité pour manier des fichiers immenses ce qui a permis aux entreprises, comme à l'administration, une gestion rapide et efficace.

Pour ce faire, il a fallu concevoir des algorithmes très astucieux et d'une grande fiabilité.

Ceux qui trient les données sont fondamentaux car ils permettent l'accessibilité d'information dans des délais très brefs. Aussi, s'est-on attelé à concevoir ceux qui nécessitent le moins d'opérations. La scientifique américaine Betty Holberton (1917-2001) en a été une des pionnières.

Le mathématicien et informaticien américain Edgar Franck Codd (1923-2003) a conçu une théorie mathématique pour traiter de la manière la plus efficiente les bases de données, appelée modèle relationnel. Elle consiste en une modélisation des relations entre les données et en la manière de les ordonner.

OBJECTIFS

Les bases de données permettent de stocker, organiser et analyser des données.

- Comprendre les concepts du modèle relationnel :
  - acquérir les notions et le vocabulaire ;
  - être capable de construire un schéma relationnel simple ;
  - déterminer les éventuelles anomalies d'un schéma.
- Assimiler la notion de clé primaire :
  - connaître le rôle d'une clé primaire et mesurer son intérêt ;
  - comprendre la nécessité d'utiliser suivant les cas un seul attribut ou plusieurs.
- Aborder le modèle entités et associations :
  - connaître différentes représentations ;
  - être capable de décrire différents types d'associations ;
  - établir les liens avec les notions de clé primaire et de clé étrangère.

Cours

Introduction

Les données en général et la gestion de ces données sont omniprésentes dans de nombreuses activités. La quantité de données est gigantesque et se mesure en gigaoctet, téraoctet ou pétaoctet. L'utilisation de bases de données relationnelles permet le stockage de données ainsi que l'organisation des mises à jour et des consultations par un grand nombre d'utilisateurs.

Principes et architecture

- Les données sont stockées physiquement dans une machine.
- Le modèle relationnel représente la logique de l'organisation des données.
- L'utilisateur devant sa machine ne se préoccupe ni de l'aspect physique, ni de l'aspect logique.

Ces trois niveaux, physique, logique et externe, sont indépendants les uns des autres. Ainsi, une modification sur le schéma physique par exemple n'a aucune conséquence sur le schéma logique et les schémas externes.

Types de données

Les données traitées peuvent être de formats différents : des nombres, des textes, des images, des vidéos. Pour les stocker, il faut choisir comment les coder afin de les enregistrer sur un support numérique, donc définir un type de données, par exemple des types numériques, des types textes, des types dates. La manière de les stocker doit permettre d'accéder à des informations et d'effectuer des opérations simples comme un tri le plus rapidement possible.

Chaque système de gestion de bases de données, ou SGBD, présente quelques différences sur les types disponibles mais en propose suffisamment afin de pouvoir utiliser dans chaque cas le plus économique en besoin au niveau de la mémoire et limiter ainsi la taille de la base de données.

Les types numériques se décomposent en types entiers ou pas. Il y a en général plusieurs types prévus pour les entiers (selon la taille et le signe), mais pour la suite on se contente d'envisager un seul type *entier*.

Pour les nombres non entiers plusieurs types peuvent aussi être proposés mais on se limite dans la suite au type *flottant*.

Une représentation des nombres en machine a été étudiée en première année avec des mots de taille fixe (8, 16, 32, 64 bits). Les SGBD peuvent utiliser des représentations différentes pour certains types de nombres afin de pouvoir les stocker de manière exacte et effectuer des calculs exacts. Par exemple, pour un nombre décimal, on peut envisager de coder en binaire chaque chiffre de son écriture. Avec des mots de même taille pour chaque chiffre, nous avons besoin de quatre bits par chiffre. Par exemple, le nombre 68,3 est codé par 0110 1000 0011. L'exactitude nécessite plus de place en mémoire.

Pour les textes également on peut disposer de plusieurs types principalement liés à la taille maximale que l'on prévoit d'utiliser, mais dans la suite on se restreint simplement à un type *chaîne de caractères*.

Concernant les dates, il existe des types précis mais on se limite à des chaînes de caractères dans le format `aaa-mm-jj`, ce qui est suffisant pour pouvoir effectuer des comparaisons en utilisant l'ordre

lexicographique. On peut aussi utiliser des nombres entiers représentant le nombre de secondes écoulées depuis une date origine qui est classiquement le 1er janvier 1970.

■ Modèle relationnel

L'ensemble des informations à stocker doit être analysé afin d'être modélisé. Ensuite il s'agit de concevoir la base de données. D'une certaine manière, cela signifie implémenter le modèle.

Une différence essentielle avec l'utilisation d'un tableur pour stocker des données et effectuer des calculs avec ces données apparaît ici : la notion de modélisation. C'est cette modélisation qui permet d'avoir des liens entre les données, de les contrôler (sécurité, validité), de les traiter à l'aide d'un programme informatique, de les rendre accessibles simultanément à un grand nombre d'utilisateurs, de pouvoir gérer des volumes de données de très grande taille.

Pour modéliser des relations entre plusieurs informations et les ordonner entre elles, une réflexion préalable sur les traitements qu'on souhaite réaliser sur les données est essentielle et déterminante. Les principaux concepts qui interviennent sont la *relation*, le *domaine*, l'*identifiant* ou *clé primaire*, l'*identifiant externe* ou *clé étrangère*.

Remarque : en langage des bases de données, on parle de *table* plutôt que de relation et de clé primaire plutôt que d'identifiant. Dans la suite les termes sont interchangeables.

■ Un point d'histoire

Edgar Frank Codd surnommé Ted Codd, qui a reçu le prix Turing en 1981, travaillait au laboratoire IBM en 1970 et proposa un modèle logique pour gérer le partage de gros volumes de données. Le premier système vit le jour dix ans plus tard avec la société Oracle. Des notions mathématiques comme le langage ensembliste, les relations, et la logique sont à la base de son modèle.

■ Schéma relationnel

Les relations

Considérons dans un lycée des données concernant les élèves, les professeurs, les administratifs. On représente l'ensemble des données, comme le nom, le prénom, la date de naissance, le numéro de téléphone de chaque personne, dans un tableau à deux dimensions appelé *relation* ou *table*.

Une colonne du tableau est appelée *attribut* ou *champ*, par exemple la colonne nommée `nom`. L'ensemble des colonnes caractérisent la relation.

Les valeurs des attributs pour une personne particulière constituent une ligne qui est appelée *enregistrement* ou *n-uplet*, (*tuple* en anglais), ou *instance*. Chaque n-uplet est unique. Chaque valeur a un type, le même type pour toutes les valeurs contenues dans une colonne, au sens type de donnée en programmation (entier, texte, flottant, ...), avec des valeurs limites (par exemple les entiers naturels inférieurs à 256, les textes comportant au maximum 8 caractères, etc). On dit que les valeurs permises dans une colonne appartiennent à un *domaine*.

L'obligation d'appartenance d'une valeur à un domaine s'appelle une *contrainte d'intégrité*. Plus précisément, il s'agit ici d'une *intégrité de domaine*.

Le *produit cartésien* des domaines D_1, D_2, ..., D_n noté $D_1 \times D_2 \times \ldots \times D_n$ est l'ensemble des n-uplets $(v_1, v_2, \ldots, v_n)$ tels, pour tout i, v_i est un élément de D_i.

Une relation est donc un sous-ensemble du produit cartésien de domaines. Elle est caractérisée par un nom.

Par exemple, supposons donnés les domaines $D_1 = \{a, b, c\}$ et $D_2 = \{x, y\}$.

Alors : $D_1 \times D_2 = \{(a,x),(a,y),(b,x),(b,y),(c,x),(c,y)\}$.

D_1 \ D_2	x	y
a	(a,x)	(a,y)
b	(b,x)	(b,y)
c	(c,x)	(c,y)

Une relation est par exemple : $R = \{(a,x),(b,x),(c,y)\}$.

La relation R contient trois enregistrements ou trois lignes. On peut donner un nom à chacune des deux colonnes, la première qui contient des valeurs appartenant au domaine D_1, et la seconde qui contient des valeurs appartenant au domaine D_2

■ Notion de clé primaire

- Dans toute relation, un attribut, ou un groupe d'attributs, doit permettre d'identifier de manière unique un enregistrement. On l'appelle une *clé candidate*.
- S'il y a plusieurs clés candidates, on en privilégie une, nommée la *clé primaire*. Choisir une clé primaire est une contrainte d'intégrité imposée à la relation.
- Certaines relations peuvent contenir un attribut dont les valeurs permises sont uniquement celles prises par la clé primaire d'une autre relation. Cet attribut est alors appelé *clé étrangère*. Une clé étrangère est une *contrainte d'intégrité référentielle* (elle fait référence à un attribut d'une autre table). Elle relie deux tables de manière cohérente.

Les clés primaires et étrangères sont utilisées dans le modèle entité-association présenté dans la suite. Elles permettent de spécifier des contraintes dans les bases de données et d'effectuer des opérations de jointures entre les tables, opérations présentées dans le chapitre suivant.

La présence d'une clé primaire est capitale pour assurer que les n-uplets sont tous distincts. Une contrainte d'unicité est donc imposée : chaque valeur prise par l'attribut, ou le groupe d'attributs, sur lequel est posé une contrainte de clé primaire est unique.

De manière générale, une contrainte d'intégrité permet d'assurer une certaine cohérence dans les données. Il est par exemple impossible qu'une valeur de l'attribut clé étrangère d'une relation ne soit pas une valeur de l'attribut clé primaire de l'autre relation. Il est aussi impossible de supprimer une valeur de la clé primaire qui est une valeur de la clé étrangère.

■ Exemples

Considérons une base de données qui sert à stocker des informations dans un établissement scolaire. Pour représenter le concept d'élève, nous utilisons une relation `Élève`. Les attributs sont par exemple nommés `INE` (numéro d'identification), `Nom`, `Prénom`, `Âge`, `Sexe`, `Classe`. Chaque élève est représenté par un enregistrement. Une deuxième relation `Classe` représente le concept de classe, avec pour attributs `Nom`, `Effectif`, `Salle`. Il y a ensuite des liens entre l'ensemble des élèves et l'ensemble des classes. Un élève donné fait partie d'une classe particulière. Ce lien fonctionne dans les deux sens : une classe contient plusieurs élèves.

Les données peuvent être modélisées d'une autre manière. On ajoute une relation `Personne` pour représenter les personnes appartenant à la communauté scolaire. Si cette relation dispose d'attributs comme le nom, le prénom, etc., on peut alors supprimer ces attributs de la relation `Élève` et ne garder que le numéro INE et la classe. Pour des relations comme `Professeur`, `Administratif`,

`Service`, on peut remplacer l'attribut INE par un attribut ayant pour valeur le numéro de sécurité sociale par exemple.

Méthode

On définit un *schéma de relation*. Pour cela, on donne le nom de la relation, on précise le nom des attributs et pour chaque attribut le domaine de valeurs. Ce domaine est défini par un type et par des restrictions. Le domaine est une contrainte d'intégrité qui est garantie par le SGBD.

Préciser quelles sont les valeurs admises pour un attribut, le type de ces valeurs et les bornes, permet d'ajouter du sens. Par exemple, décider que le nom est une chaîne de caractères avec un nombre maximal de caractères ou que le sexe est un caractère "F" ou "M".

Un schéma de relation se présente sous cette forme :
`Relation (attribut 1, attribut 2, ..., attribut N)`
Clé primaire : `attribut 1`
Clé étrangère : `attribut N` en référence à la clé primaire d'une autre relation

Par exemple :
`Élève (INE, Nom, Prénom, Âge, Sexe, Classe)`
Les attributs `INE` et `Âge` sont de type entier, les autres de type chaîne de caractères.
Clé primaire : `INE`
Clé étrangère : `Classe` en référence à la clé primaire d'une autre relation.

Ce schéma peut aussi se noter :
`Élève (INE, Nom, Prénom, Âge, Sexe, #Classe)`
L'attribut sur lequel est posé une contrainte de clé primaire est souligné et l'attribut sur lequel est posé une contrainte de clé étrangère est précédé du caractère `#`.

On peut représenter le schéma ainsi :

Élève
INE
Nom
Prénom
Âge
Sexe
#Classe

Un schéma relationnel, ou schéma d'une base de données relationnelle, est un ensemble de schémas de relations liés par des attributs communs.

Conception d'une relation

Si à une valeur d'un attribut A correspond une et une seule valeur d'un attribut B, on dit que l'attribut B est en *dépendance fonctionnelle* de l'attribut A.

Par exemple, à un identifiant d'élève correspond un unique nom d'élève.

Dans l'élaboration des relations, il est important d'éviter la redondance des données et de faciliter leur mise à jour. Cela permet de garantir la cohérence des informations.

De manière générale, tous les attributs ne contiennent qu'une seule information et sont en dépendance fonctionnelle de la clé primaire. On n'utilisera pas, par exemple, un attribut `Nom Prénom` qui contient deux informations.

Si une clé primaire est constituée de plusieurs attributs, les autres attributs sont en dépendance fonctionnelle de l'intégralité de la clé primaire et pas seulement d'une partie de celle-ci.

De plus, tous les attributs sont en dépendance fonctionnelle uniquement de la clé primaire.

Organisation des données

Les données peuvent être représentées dans un tableau comme celui-ci :

INE	Nom	Prénom	Âge	Sexe	Classe
1	Turing				2
2	Hopper				4
3	Codd				1
...					
50	Bachman				
...					

Remarque : ce tableau ne doit pas être interprété comme celui obtenu dans un tableur.

Un tableur permet d'effectuer diverses actions sur ces données : des calculs, des schémas, des graphiques. C'est principalement un moyen de stocker ces données. On l'utilise d'ailleurs pour recueillir des informations provenant d'une base de données ou pour communiquer des informations à une base de données. Mais il n'y a aucune modélisation, pas de liens entre les données. Un texte peut être remplacé par un nombre, des données peuvent être modifiées ou supprimées, il n'y a pas d'accès multi-utilisateurs. Ces questions ne sont pas gérées par le tableur.

Chaque case de ce tableau contient une information unique. Par exemple, on n'inscrit pas deux prénoms dans une case de la colonne `Prénom`.

L'attribut `INE` est un identifiant numérique qui permet d'identifier chaque ligne de manière unique. En effet, on n'est pas certain de pouvoir identifier une ligne uniquement à l'aide du nom, car plusieurs élèves peuvent avoir le même nom, ni même en utilisant le nom et le prénom.

Cet attribut `INE` peut donc être défini comme clé primaire. On dit qu'on pose *une contrainte de clé primaire sur l'attribut `INE`*.

Pour la deuxième relation nommée `Classe`, on a le tableau :

Nom	Effectif	Salle
PSI1	45	B307
PC	38	C206
PSI2	42	B311
PT1	35	B205
...	...	...

Deux tableaux représentent les données qui avec un tableur pourraient être stockées dans un tableau unique.

Pour distinguer les attributs `Nom` des relations `Élève` et `Classe`, nous pouvons utiliser la notation pointée comme `Élève.Nom` et `Classe.Nom`.

L'attribut `Nom` de la relation `Classe` est une clé primaire. Nous souhaitons poser une contrainte de clé étrangère sur l'attribut `Classe` de la relation `Élève`.

Or, pour effectuer des recherches et pour procéder à d'éventuelles modifications, il est plus efficace d'utiliser un nombre entier plutôt qu'une chaîne de caractères. On ajoute donc une colonne

supplémentaire, un nouvel attribut nommé `Id`. Il s'agit d'une clé artificielle ou clé de substitution (*surrogate key* en anglais) qui permet d'identifier une ligne quelconque.

Id	Nom	Effectif	Salle
1	PSI1	45	B307
2	PC	38	C206
3	PSI2	42	B311
4	PT1	35	B205
...	...	...	...

Les valeurs de la clé étrangère `Classe` de la relation `Élève` sont alors des valeurs de la clé primaire `Id` de la relation `Classe`.

Avec cette représentation, les modifications sont beaucoup plus simples à gérer que si toutes les données étaient dans un tableau unique. Ainsi, si par exemple les élèves de la classe PSI1 changent de salle, il n'y a qu'une seule valeur à modifier dans le tableau `Classe`.

Remarque : une relation est bien autre chose qu'un simple tableau. Nous pouvons permuter les colonnes ou les lignes sans modifier la relation qui est un ensemble de n-uplets. Par exemple la relation $R = \{(a, x), (b, x), (c, y)\}$ peut aussi s'écrire $R = \{(x, b), (y, c), (x, a)\}$.

La notion de contrainte est capitale. Supposons que dans la table `Classe` un enregistrement, ou une ligne, contient la valeur 7 pour l'identifiant `Id` et que cette valeur 7 est celle d'un ou plusieurs enregistrements pour l'attribut `Classe` dans la table `Élève`.

Si par erreur, la suppression dans la table `Classe` de l'enregistrement avec l'identifiant 7 est demandée, cela va provoquer une erreur car cela conduirait à une violation de la contrainte de référence, ou contrainte de clé étrangère, de `Élève.Classe` vers `Classe.Id`.

En pratique, c'est généralement le comportement par défaut. Mais il peut être modifié en imposant par exemple une suppression en cascade. Dans ce cas, toutes les lignes concernées de la table `Élève` sont supprimées. On peut aussi imposer que les lignes concernées ne soient pas supprimées mais qu'une valeur par défaut remplace la valeur 7.

■ Bases de données relationnelles et SGBD

Dans une base de données, les informations doivent pouvoir être facilement consultables et mises à jour. Il existe plusieurs types de bases de données. Une base de données relationnelle utilise le modèle relationnel. Un SGBD est un système de gestion de base de données. L'informaticien américain Charles Bachman est le premier concepteur d'un SGBD moderne.

■ SGBD

Pour accéder à une base de données, on utilise donc un SGBD. Ce peut être un composant logiciel, comme SQLite qui est une bibliothèque avec une interface directement intégrable dans un programme. D'autres SGBD, comme MySQL et PostgreSQL, fonctionnent avec un serveur.

Les systèmes de gestion de base de données sont de deux types :

- des systèmes libres comme MySQL, PostgreSQL, SQLite ;
- des systèmes propriétaires comme Oracle Database de l'entreprise Oracle, SQL Server de Microsoft, Access qui fait partie de la suite Microsoft Office.