Support de cours

BASE DE DONNEES ET SYSTEMES DE GESTION DE BASE DE DONNEES (SGBD)

CHAPITRE 1 – GENERALITES (Suite)

Présenté par Anani ATADEGNON

S O M M A I R E

I. Objectifs.....3
II. Définitions.....3
III. Caractéristiques d'un SGBD.....3
IV. Modèles de description des données.....4
V. Système de Gestion de Bases de Données (SGBD).....6
V.1. Principes de fonctionnement.....6
V.2. Les Langages dans les SGBD.....7
V.3. Niveaux de description des données ANSI / SPARC.....7
V.4. Exemples d'application des Bases de Données et de SGBD.....7
VI. Modèle relationnel.....8
VII. Opérateurs relationnels.....8
VIII. Formes normales.....9
VIII.1. Dépendances fonctionnelles.....9
VIII.2. Modèle Conceptuel des Données (MCD).....12
VIII.3. Règles relatives au MCD : Normalisation des entités.....13
IX. Langages de manipulation de données relationnelles.....15
Conclusion.....16

I. Objectifs

Ce document a pour objectifs de répondre aux questions suivantes :

- Qu'est-ce donc qu'une base de données ?
- Que peut-on attendre d'un système de gestion de bases de données ?

II. Définitions

1. **BASE DE DONNEES :** Une base de données est un ensemble d'informations sur un sujet qui est : exhaustif, non redondant, structuré.
2. **SYSTEME DE GESTION DE BASE DE DONNEES (SGBD) :** Un système de gestion de base de données est un logiciel qui permet de décrire, modifier, interroger, administrer, les données d'une base de données.

   Les principaux systèmes de gestion de bases de données sont les suivants : Borland Paradox, Filemaker, IBM DB2, Ingres, Interbase, Microsoft SQL server, Microsoft Access, Microsoft FoxPro, Oracle, Sybase, MySQL, PostgreSQL, mSQL, SQL Server, etc.

III. Caractéristiques d'un SGBD

L'architecture à trois niveaux définie par le standard ANSI/SPARC permet d'avoir une indépendance entre les données et les traitements. D'une manière générale un SGBD doit avoir les caractéristiques suivantes :

- **Indépendance physique :** Le niveau physique peut être modifié indépendamment du niveau conceptuel. Cela signifie que tous les aspects matériels de la base de données n'apparaissent pas pour l'utilisateur, il s'agit simplement d'une structure transparente de représentation des informations.

  La façon dont les données sont définies doit être indépendante des structures de stockage utilisées.

- **Indépendance logique :** le niveau conceptuel doit pouvoir être modifié sans remettre en cause le niveau physique, c'est-à-dire que l'administrateur de la base doit pouvoir la faire évoluer sans que cela gêne les utilisateurs.

  Un même ensemble de données peut être vu différemment par des utilisateurs différents. Toutes ces visions personnelles des données doivent être intégrées dans une vision globale.

- **Manipulabilité :** des personnes ne connaissant pas la base de données doivent être capables de décrire leurs requêtes sans faire référence à des éléments techniques de la base de données.

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

- **Rapidité des accès :** le système doit pouvoir fournir les réponses aux requêtes le plus rapidement possible, cela implique des algorithmes de recherche rapides.

- **Administration centralisée des données (intégration) :** le SGBD doit permettre à l'administrateur de pouvoir manipuler les données, insérer des éléments, vérifier son intégrité de façon centralisée.

  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.

- **Limitation de la redondance :** le SGBD doit pouvoir éviter dans la mesure du possible des informations redondantes, afin d'éviter d'une part un gaspillage d'espace mémoire mais aussi des erreurs.
- **Non-redondance des données :** Afin d'éviter les problèmes lors des mises à jour, chaque donnée ne doit être présente qu'une seule fois dans la base.
- **Vérification de l'intégrité ou cohérence des données :** les données doivent être cohérentes entre elles, de plus lorsque des éléments font références à d'autres, ces derniers doivent être présents.

  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 des données. Les contraintes d'intégrité sont décrites dans le Langage de Description de Données (LDD).
- **Partageabilité des données :** le SGBD doit permettre l'accès simultané à la base de données par plusieurs utilisateurs.

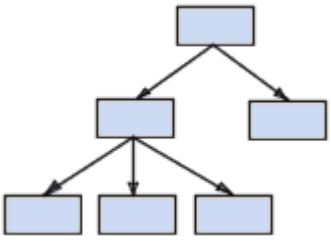
  Il s'agit de permettre à plusieurs utilisateurs d'accéder aux mêmes données au même moment de manière transparente. Si ce problème est simple à résoudre quand il s'agit uniquement d'interrogations, cela ne l'est plus quand il s'agit de modifications dans un contexte multiutilisateur, car il faut : permettre à deux (ou plus) utilisateurs de modifier la même donnée « en même temps » et assurer un résultat d'interrogation cohérent pour un utilisateur consultant une table pendant qu'un autre la modifie.
- **Sécurité des données :** Le SGBD doit présenter des mécanismes permettant de gérer les droits d'accès aux données selon les utilisateurs.

  Les données doivent pouvoir être protégées contre les accès non autorisés. Pour cela, il faut pouvoir associer à chaque utilisateur des droits d'accès aux données.
- **Résistance aux pannes :** Que se passe-t-il si une panne survient au milieu d'une modification, si certains fichiers contenant les données deviennent illisibles ? Il faut pouvoir récupérer une base dans un état « sain ». Ainsi, après une panne intervenant au milieu d'une modification, deux solutions sont possibles : soit récupérer les données dans l'état dans lequel elles étaient avant la modification, soit terminer l'opération interrompue.

IV. Modèles de description des données

Les bases de données sont apparues à la fin des années 1960, à une époque où la nécessité d'un système de gestion de l'information souple se faisait ressentir. Ces premiers systèmes sont uniquement basés sur la gestion de fichiers plus ou moins sophistiqués. Il existe quatre modèles de SGBD, différenciés selon la représentation des données qu'elle contient :

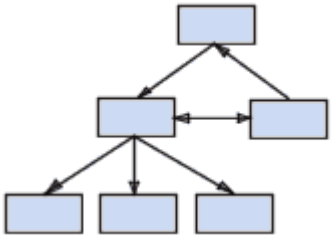
- **le modèle hiérarchique :** les données sont classées hiérarchiquement, selon une arborescence descendante. Ce modèle utilise des pointeurs entre les différents enregistrements. Il s'agit du premier modèle de SGBD.



Une base de données hiérarchique est une forme de système de gestion de base de données qui lie des

enregistrements dans une structure arborescente de façon à ce que chaque enregistrement n'ait qu'un seul possesseur (par exemple, une paire de chaussures n'appartient qu'à une seule personne). Les structures de données hiérarchiques ont été largement utilisées dans les premiers systèmes de gestion de bases de données conçus pour la gestion des données du programme Apollo de la NASA. Cependant, à cause de leurs limitations internes, elles ne peuvent pas souvent être utilisées pour décrire des structures existantes dans le monde réel. Les liens hiérarchiques entre les différents types de données peuvent rendre très simple la réponse à certaines questions, mais très difficile la réponse à d'autres formes de questions. Si le principe de relation « 1 vers N » n'est pas respecté (par exemple, un malade peut avoir plusieurs médecins et un médecin a, *a priori*, plusieurs patients), alors la hiérarchie se transforme en un réseau.

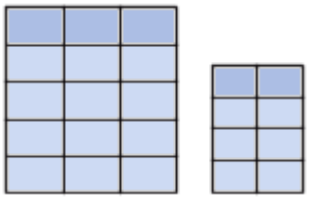
- **le modèle réseau :** Comme le modèle hiérarchique ce modèle utilise des pointeurs vers des enregistrements. Toutefois la structure n'est plus forcément arborescente dans le sens descendant.



Le modèle réseau est en mesure de lever de nombreuses difficultés du modèle hiérarchique grâce à la possibilité d'établir des liaisons de type n-n, les liens entre objets pouvant exister sans restriction. Pour retrouver une donnée dans une telle modélisation, il faut connaître le chemin d'accès (les liens) ce qui rend les programmes dépendants de la structure de données

Les modèles **hiérarchique et réseau** sont apparus en 1970. Ce modèle de bases de données a été inventé par C.W. Bachman. Pour son modèle, il reçut en 1973 le prix Turing.

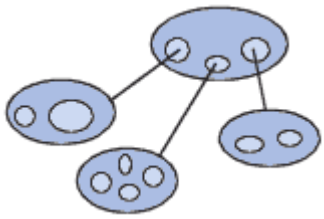
- **le modèle relationnel (SGBDR, Système de gestion de bases de données relationnelles) :** les données sont enregistrées dans des tableaux à deux dimensions (lignes et colonnes). La manipulation de ces données se fait selon la théorie mathématique des relations (Algèbre relationnelle). Ce modèle apparaît sur le marché en 1980.



Une base de données relationnelle est une base de données structurée suivant les principes de l'algèbre relationnelle. Le père des bases de données relationnelles est *Edgar Frank Codd*. Chercheur chez IBM à la fin des années 1960, il étudiait alors de nouvelles méthodes pour gérer de grandes quantités de données, car les modèles et les logiciels de l'époque ne le satisfaisaient pas. Mathématicien de formation, il était persuadé qu'il pourrait utiliser des branches spécifiques des mathématiques (la théorie des ensembles et la logique des prédicats du premier ordre) pour résoudre des difficultés telles que la redondance des données, l'intégrité des données ou l'indépendance de la structure de la base de données avec sa mise en œuvre physique. En 1970, il publia un article où il proposait de stocker des données hétérogènes dans des tables, permettant d'établir des relations entre elles. De nos jours, ce modèle est extrêmement répandu, mais en 1970, cette idée était considérée comme une curiosité intellectuelle. On doutait que les tables puissent être jamais gérées de manière efficace par un ordinateur. Ce scepticisme n'a cependant pas empêché Codd de poursuivre ses recherches. Un premier prototype de Système de gestion de bases de données relationnelles (SGBDR) a été construit dans les laboratoires d'IBM. Depuis

les années 1980, cette technologie a mûri et a été adoptée par l'industrie. En 1987, le langage SQL, qui étend l'algèbre relationnelle, a été standardisé. C'est dans ce type de modèle que se situe ce cours de base de données

- **le modèle objet (SGBDO, Système de gestion de bases de données objet) :** les données sont stockées sous forme d'objets, c'est-à-dire de structures appelées classes présentant des données membres. Les champs sont des instances de ces classes.



La notion de bases de données objet ou relationnel-objet est plus récente et encore en phase de recherche et de développement. Elle sera très probablement ajoutée au modèle relationnel.

A la fin des années 1990, les bases relationnelles sont les bases de données les plus répandues : (environ trois quarts des bases de données).

V. 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 base de données (SGBD). Un SGBD est caractérisé par le modèle de description des données qu'il supporte (hiérarchique, réseau, relationnel, objet). Les données sont décrites sous la forme de ce modèle, grâce à un Langage de Description des Données (LDD). Cette description est appelée schéma. Une fois la base de données spécifiée, on peut y insérer des données, les récupérer, les modifier et les détruire. C'est ce qu'on appelle manipuler les données. Les données peuvent être manipulées non seulement par un Langage spécifique de Manipulation des Données (LMD), mais aussi par des langages de programmation classiques.

V.1. Principes de fonctionnement

Un SGBD doit permettre l'ajout, la modification et la recherche de données. Un système de gestion de bases de données héberge généralement plusieurs bases de données, qui sont destinées à des logiciels ou des thématiques différents. Actuellement, la plupart des SGBD fonctionnent selon un mode client/serveur. Le serveur (sous-entendu la machine qui stocke les données) reçoit des requêtes de plusieurs clients et ceci de manière concurrente. Le serveur analyse la requête, la traite et retourne le résultat au client.

Quel que soit le modèle, un des problèmes fondamentaux à prendre en compte est la cohérence des données. Par exemple, dans un environnement où plusieurs utilisateurs peuvent accéder concurremment à une colonne d'une table par exemple pour la lire ou pour l'écrire, il faut s'accorder sur la politique d'écriture. Cette politique peut être : les lectures concurrentes sont autorisées, mais dès qu'il y a une écriture dans une colonne, l'ensemble de la colonne est envoyé aux autres utilisateurs l'ayant lue pour qu'elle soit rafraîchie.

V.2. Les Langages dans les SGBD

Pour pouvoir être utilisable, le SGBD doit communiquer avec les utilisateurs à l'aide d'un langage. En pratique, on distingue :

Le langage de définition des données (LDD) : Il permet de décrire les entités et les associations qui vont constituer les schémas conceptuels, externes et internes de la base.

Le langage de manipulation des données (LMD) : Pour décrire les programmes d'applications (consultation, mise à jour des données)

Le langage d'interrogation (généralement considéré comme un sous ensemble du LMD) : Permet d'interroger directement la base, à l'aide des commandes simples sans recours à un programme.

Le langage de contrôle de données (LCD) : Permet le contrôle d'accès aux données et l'intégrité des données.

V.3. Niveaux de description des données ANSI / SPARC

L'architecture ANSI/SPARC, datant de 1975, définit des niveaux pour un système de gestion de bases de données, on distingue :

- **Le niveau externe** : correspond à la perception de tout ou partie de la base par un groupe donné d'utilisateurs, indépendamment des autres. On appelle cette description le *schéma externe* ou *vue*. Il peut exister plusieurs schémas externes représentant différentes vues sur la base de données avec des possibilités de recouvrement. Le niveau externe assure l'analyse et l'interprétation des requêtes en primitives de plus bas niveau et se charge également de convertir éventuellement les données brutes, issues de la réponse à la requête, dans un format souhaité par l'utilisateur. Il définit les vues des utilisateurs.
- **Le niveau conceptuel** : décrit la structure de toutes les données de la base, leurs propriétés (*i.e.* les relations qui existent entre elles : leur sémantique inhérente), sans se soucier de l'implémentation physique ni de la façon dont chaque groupe de travail voudra s'en servir. Dans le cas des SGBD relationnels, il s'agit d'une vision tabulaire où la sémantique de l'information est exprimée en utilisant les concepts de relation, attributs et de contraintes d'intégrité. On appelle cette description le *schéma conceptuel*. Il est appelé aussi MCD (modèle conceptuel des données) ou MLD (modèle logique des données). Il définit l'arrangement des informations au sein de la base de données.
- **Le niveau interne ou physique** : s'appuie sur un système de gestion de fichiers pour définir la politique de stockage ainsi que le placement des données. Le niveau physique est donc responsable du choix de l'organisation physique des fichiers ainsi que de l'utilisation de telle ou telle méthode d'accès en fonction de la requête. On appelle cette description le *schéma interne*. Il définit la façon selon laquelle sont stockées les données et les méthodes pour y accéder,

V.4. Exemples d'application des Bases de Données et de SGBD

Les données étant omniprésentes autour de nous, les bases de données le sont aussi. Nous pouvons citer entre autres :

- *Base de données d'une école ;*
- *Base de données d'un hôpital ;*
- *Base de données bibliothécaire ;*
- *Base de données sur des films (IROKO +, etc.) ;*
- *Réservation de place d'avions ;*
- *Communications téléphoniques ;*
- *Sécurité sociale (INAM, etc.)*

- *Système d'Information Géographique,*
- *Système de messagerie d'une entreprise, etc.*

Il existe de nombreux systèmes de gestion de bases de données, en voici une liste non exhaustive :

- *PostgreSQL ;*
- *MySQL ;*
- *Oracle ;*
- *IBM DB2 ;*
- *Microsoft SQL ;*
- *Sybase ;*
- *Informix.*

VI. Modèle relationnel

Le modèle relationnel a été formalisé par CODD en 1970. Quelques exemples de réalisation en sont : DB2(IBM), INFORMIX, INGRES, ORACLE, mSQL, MySQL, etc.

Dans ce modèle, les données sont stockées dans des tables, sans préjuger de la façon dont les informations sont stockées dans la machine. Un ensemble de données sera donc modélisé par un ensemble de tables.

Le succès du modèle relationnel auprès des chercheurs, concepteurs et utilisateurs est dû à la puissance et à la simplicité de ses concepts. En outre, contrairement à certains autres modèles, il repose sur des bases théoriques solides, notamment la théorie des ensembles et la logique mathématique.

Les objectifs du modèle relationnel :

- proposer des schémas de données faciles à utiliser,
- améliorer l'indépendance logique et physique,
- mettre à la disposition des utilisateurs des langages de haut niveau pouvant éventuellement être utilisés par des non informaticiens,
- optimiser les accès à la base de données,
- améliorer l'intégrité et la confidentialité,
- fournir une approche méthodologique dans la construction des schémas.

De façon informelle, on peut définir le modèle relationnel de la manière suivante :

- Les données sont organisées sous forme de tables à deux dimensions, encore appelées relations et chaque ligne n-uplet ou tuple,
- les données sont manipulées par des opérateurs de l'algèbre relationnelle,
- l'état cohérent de la base est défini par un ensemble de contraintes d'intégrité.

Au modèle relationnel est associée la théorie de la normalisation des relations qui permet de se débarrasser des incohérences au moment de la conception d'une base de données.

VII. Opérateurs relationnels

Définition (Projection) : Opération qui consiste à supprimer des attributs (propriétés ou champs ou colonnes) d'une relation et à éliminer les tuples (ligne d'une table) en double apparaissant dans la nouvelle relation.

Exemple : la sélection de certaines colonnes d'une table est l'opération de projection ;

Définition (Restriction) : Opération qui consiste à supprimer les tuples d'une relation ne satisfaisant

pas la condition précisée.

Exemple : de sélectionner certaines lignes d'une table en fonction de leur contenu : c'est l'opération de restriction ;

Définition (Jointure) : Opération qui consiste à faire le produit cartésien de deux relations, puis à supprimer les tuples ne satisfaisant pas une condition portant sur un attribut de la première relation et sur un attribut de la seconde.

Définition (Union) : Opération portant sur deux relations ayant le même schéma et construisant une troisième relation constituée des tuples appartenant à chaque relation. Les tuples en double sont éliminés.

Définition (Différence relationnelle) : Opération portant sur deux relations ayant le même schéma et construisant une troisième relation dont les tuples sont constitués de ceux ne se trouvant que dans une seule relation.

Définition (Intersection) : Opération portant sur deux relations ayant le même schéma et construisant une troisième relation dont les tuples sont constitués de ceux appartenant aux deux relations.

Remarques : La combinaison des informations venant de plusieurs tables sont les opérations de jointure, union, intersection, différence relationnelle ;

VIII. Formes normales

VIII.1. Dépendances fonctionnelles

DEPENDANCE FONCTIONNELLE : on dit que deux propriétés **a et b** sont reliées par une dépendance fonctionnelle si la connaissance de la valeur de **a** détermine une et une seule valeur de **b**.

a ——— d f ⟶ b

Exemple :

Code – client ——— df ⟶ nom-client

La connaissance du code-client détermine une et une seule valeur de nom-client. Autrement dit, si on connaît le code du client, on doit pouvoir connaître son nom et celui-ci sera unique.

La réciproque est fausse. Le nom du client ne permet pas de déterminer son code, car plusieurs clients peuvent avoir le même nom.

Nom-client ——— df ⟶ code-client n'est pas vrai.

La dépendance fonctionnelle peut porter sur la concaténation de plusieurs propriétés.

Exemple :

N° -BON-DE-CDE + REF. ——— df ⟶ QTE COMMANDEE

La référence seule ne suffit pas à déterminer la quantité commandée, une même référence pouvant figurer sur plusieurs bons de commande.

Le numéro de BON DE COMMANDE ne suffit pas non plus puisqu'un même BON BON DE COMMANDE peut concerner plusieurs références.

En revanche, si on admet qu'une référence ne peut figurer qu'une seule fois sur un bon, la connaissance du numéro du BON DE COMMANDE et de la référence détermine celle de la quantité commandée.

DEPENDANCE FONCTIONNELLE ELEMENTAIRE : on dit qu'il y a dépendance fonctionnelle élémentaire entre les propriétés a et b et on note :

a ⟶ b

si a ——— df ⟶ b et si aucune partie de **a** ne détermine **b**.

Exemple :

CODE-CLIENT + NOM-CLIENT ——df——→ ADRESSE-CLIENT

n'est pas élémentaire puisque la connaissance de CODE-CLIENT (partie de CODE-CLIENT + NOM-CLIENT) suffit à déterminer l'adresse.

CODE-CLIENT —— df ——→ ADRESSE-CLIENT

est élémentaire et on peut écrire :

CODE-CLIENT ——→ ADRESSE-CLIENT

De même N° -BON-DE-CDE + REF ——→QTE COMMANDE (dépendance élémentaire).

DEPENDANCE FONTIONNELLE ELEMENTAIRE DIRECTE : on dit que la propriété **b** dépend fonctionnellement de **a** par une dépendance fonctionnelle élémentaire directe si cette dépendance est élémentaire

a ——→ b et s'il n'existe pas de propriété c telle que :

a —— df ——→ c et c —— df ——→ b

Autrement dit on élimine toute transitivité.

Exemple :

N° -PROFESSEUR ——→ CODE-MATIERE

CODE-MATIERE ——→ NOM-MATIERE

N° -PROFESSEUR ——→ NOM-MATIERE

Les deux premières dépendances fonctionnelles sont directes, mais la troisième ne l'est pas en raison de la transitivité :

N° -PROFESSUR ——→ CODE-MATIERE ——→ NOM-MATIERE

CLE (D'IDENTIFICATION) D'UNE ENTITE : une clé d'une entité (objet) est une propriété (ou une concaténation de propriétés) de cette entité telle que toutes les autres propriétés de l'entité dépendant d'elle fonctionnellement et telle que ceci ne soit plus vrai pour aucune de ses parties.

Exemple :

CLIENT
Code-client nom adresse

CODE-CLIENT + NOM n'est pas une clé bien que :

CODE-CLIENT + NOM ——— DF ⟶ ADRESSE

car ceci reste vrai pour la partie CODE-CLIENT de CODE-CLIENT + NOM puisque CODE-CLIENT détermine ADRESSE

En revanche CODE-CLIENT est une clé car :

CODE-CLIENT ——— df ⟶ NOM

CODE-CLIENT ——— df ⟶ ADRESSE

et car ceci n'est plus vrai pour aucune partie de code-client.

Notons qu'une entité peut avoir plusieurs clés.

Le modèle devra prévoir celle des clés qui sera retenue comme identifiant.

Ainsi si l'entité SALARIE comporte les propriétés N° SS et MATRICULE, il s'agit de deux clés et il faudra choisir laquelle des deux sera l'identifiant.

Notons aussi que la dépendance fonctionnelle par laquelle une propriété dépend de la clé n'est pas forcément élémentaire.

Exemple :

LIGNE-DE-COMMANDE
N° BON-CDE, REF., QTE, DESIGN

N° BON-CDE + REF est bien une clé car on a :

N° BON-CDE + REF. ____ df ⟶ QTE et DESIGN

sans avoir :

N° BON-CDE ____ df ⟶ QTE et DESIGN

ni

REF. _____ df ⟶ QTE et DESIGN

Cependant REF ____ df ⟶ DESIGN. Implique que la dépendance fonctionnelle

N° BON-CDE + REF. _____ df ⟶ DESIGN n'est pas élémentaire.

Les dépendances fonctionnelles entre propriétés sont à considérer par rapport aux entités et relations.

VIII.2. Modèle Conceptuel des Données (MCD)

Le modèle conceptuel des données vise à donner une représentation stable de l'ensemble des données manipulées par notre site ainsi que les relations entre ces données. Ce modèle est basé sur les notions :

- **Entité** : c'est une représentation d'un élément matériel ou immatériel ayant un rôle dans le système que l'on désire décrire.
- **Relation ou association** : représente des liens sémantiques qui peuvent exister entre plusieurs entités.
- **Cardinalités** : ils permettent de caractériser le lien qui existe entre une entité et la relation à laquelle elle est reliée. La cardinalité d'une relation est composée d'un couple comportant une borne maximale et une borne minimale, intervalle dans lequel la cardinalité d'une entité peut prendre sa valeur :
  - la borne minimale (généralement 0 ou 1) décrit le nombre minimum de fois qu'une entité peut participer à une relation.
  - la borne maximale (généralement 1 ou n) décrit le nombre maximum de fois qu'une entité peut participer à une relation.

  Une cardinalité (1.n) signifie que chaque entité appartenant à une classe d'entité participe au moins une fois à la relation.

  Une cardinalité (0.n) signifie que chaque entité appartenant à une classe d'entité ne participe pas forcément à la relation.

- Un identifiant est un ensemble de propriétés (une ou plusieurs) permettant de désigner une et une seule entité. La définition originale est la suivante :

L'identifiant est une propriété particulière d'un objet telle qu'il n'existe pas deux occurrences de cet objet pour lesquelles cette propriété pourrait prendre une même valeur.

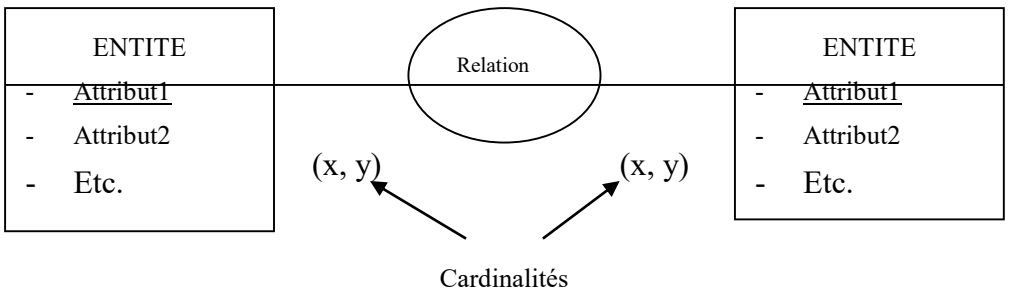
L'identifiant n'est rien d'autre que la clé d'identification.

Les attributs d'une classe d'entité permettant de désigner de façon unique chaque instance de cette entité sont appelés identifiants absolus.

Le modèle conceptuel des données propose de faire précéder d'un # les identifiants (parfois de les souligner).

Ainsi, chaque classe d'entité doit posséder au moins un attribut identifiant, et l'ensemble de ses attributs identifiants doivent être renseignés à la création de l'entité.

Le formalisme suivant représente le diagramme du MCD.



La méthode utilisée pour déduire le MCD sera la méthode entité association ou la méthode ascendante ou méthode inductive qui consiste à passer par :

- le dictionnaire des données,
- la liste des entités,
- la liste des propriétés par entité,
- la liste des relations avec leurs entités,
- les règles de gestion,
- le diagramme MCD.

VIII.3. Règles relatives au MCD : Normalisation des entités

Les entités du MCD doivent vérifier les règles suivantes :

Elémentaires et il existe au moins une **clé** caractérisant chaque occurrence de l'objet représenté.

Exemple :

Entités Normalisées

CLIENT
Nom-client, adresse-client

CLIENT
N°-client, Nom-client, Rue, ville

Cette entité n'est pas en première forme normale, car il n'y a pas de clé (plusieurs clients peuvent avoir le même nom).

D'autre part ADRESSE-CLIENT est sans doute la concaténation de RUE et VILLE et ne constitue alors pas une propriété élémentaire.

La clé si elle est unique sera prise comme identifiant. S'il y a plusieurs clés on en choisira une comme identifiant.

Toute entité doit donc avoir un identifiant.

DEUXIEME FORME NORMALE (2FN) : toute propriété d'une entité doit dépendre de la clé par une dépendance fonctionnelle élémentaire. Autrement dit toute propriété de l'entité doit dépendre de tout l'identifiant : on parle de dépendance totale.

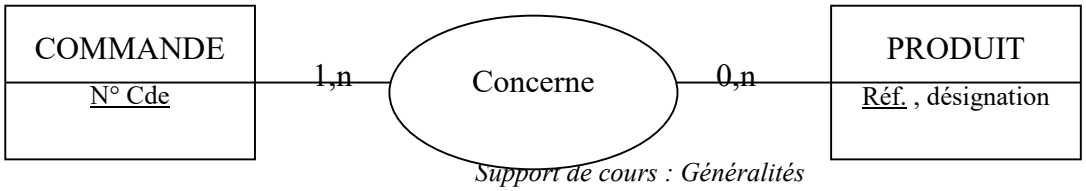
Exemple :

LIGNE-COMMANDE
N° Cde, Réf., Désignation, qté

La clé est la concaténation N° Cde + réf. Mais la dépendance fonctionnelle

N° Cde + Réf —— df ⟶ Désignation n'est pas élémentaire puisque Réf. ⟶ Désignation

Cette entité n'est pas en 2FN. Le MCD devrait devenir : 2ème forme normale



En revanche l'entité :

CLIENT
Code-client, nom, rue, ville

est en 2FN car NOM, RUE et VILLE dépendent de la clé CODE-CLIENT par une dépendance fonctionnelle élémentaire.

TROISIEME FORME NORMALE (3FN) : dans une entité toute propriété doit dépendre de la clé par une dépendance fonctionnelle élémentaire directe.

Exemple :

L'entité :

CLIENT
Code-client, nom-client, code-catégorie, nom-catégorie

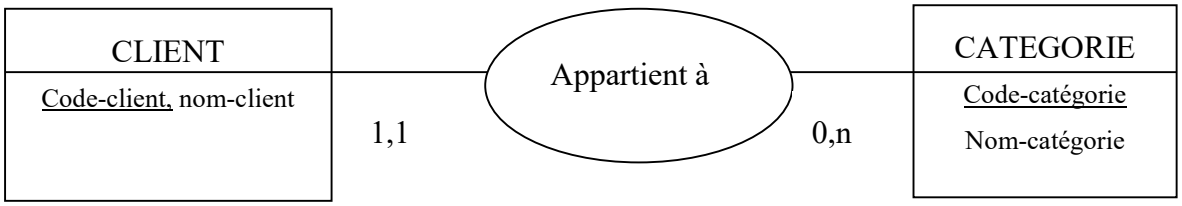
n'est pas en 3FN car la dépendance fonctionnelle :

Code-client ⟶ Nom-catégorie

n'est pas directe du fait de la transitivité :

Code-client ⟶ code-catégorie ⟶ nom-catégorie

Le MCD devrait devenir :



Les normalisations ci-dessus ont pour but d'éliminer les redondances (inutile de répéter la désignation du produit commandé à chaque commande d'un même produit) et d'éliminer les anomalies de mise à jour (si on annule un client on veut sans doute toutefois conserver la catégorie de ce client).

FORME NORMALE DE BOYCE-CODD (BCFN) : si une entité a un identifiant concaténé, un des éléments composant cet identifiant ne doit pas dépendre d'une autre propriété.

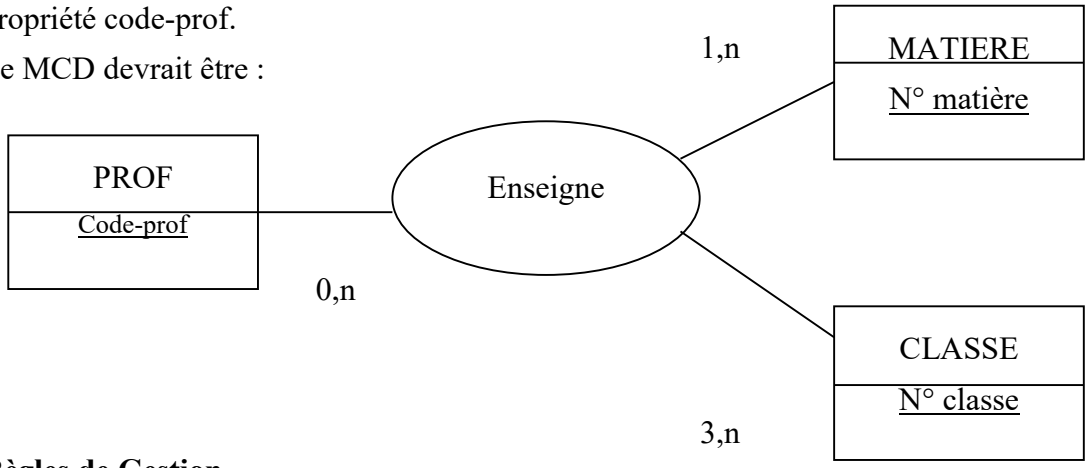
Exemple :

Soit l'entité suivante :

COURS
Matière, N° classe, code prof.

Elle n'est pas en BCFN, car Matière (partie de l'identifiant concaténé. Matière + n° classe) dépend de la propriété code-prof.

Le MCD devrait être :



Règles de Gestion

1. Tout professeur enseigne en principe au moins une matière, mais certains d'entre eux peuvent être dispensés à cause des travaux de recherche.
2. Toute matière peut être enseignée au moins dans une classe.
3. Toute classe a au moins 3 enseignements.

IX. Langages de manipulation de données relationnelles

Ces langages, dits assertionnels, sont basés sur la logique et permettent de spécifier les données que l'on souhaite obtenir, sans dire comment y accéder. On doit y trouver des opérations permettant de faire :

- **la recherche :** retrouver des tuples vérifiant certains critères,
- **l'insertion :** ajouter des tuples,
- **la suppression :** enlever des tuples vérifiant certains critères,
- **la modification :** modifier des tuples vérifiant certains critères.

Un langage de manipulation de données n'est pas utilisable à lui seul, il doit aussi pouvoir être incorporable dans un langage de programmation classique.

On peut distinguer trois grandes classes de langages :

- Les langages algébriques basés sur l'algèbre relationnelle de CODD dans lesquels les requêtes sont exprimées comme l'application des opérateurs relationnels sur des relations. C'est dans cette catégorie que l'on trouve le langage sql (*structured query language*), standard pour l'interrogation de bases de données.
- Les langages basés sur le calcul relationnel de tuples construits à partir de la logique des prédicats dans lesquels les variables manipulées sont des tuples.
- Les langages basés sur le calcul relationnel de domaines, construit aussi à partir de la logique des prédicats mais en faisant varier les variables sur les domaines des relations.

Le langage sql (Structured Query Language) comprend à lui seul l'ensemble des instructions nécessaires à la spécification et à l'utilisation d'une base de données relationnelle. C'est un langage de type déclaratif c'est-à-dire que l'on spécifie les propriétés des données que l'on recherche et pas, comme dans un langage impératif, comment les retrouver.

Le langage sql est un langage normalisé, la dernière version de la norme date de 1992.

C'est à la fois :

- un langage d'interrogation de données (LID) : SELECT ;
- un langage de manipulation de données (LMD) : UPDATE, INSERT, DELETE ;
- un langage de définition des données (LDD) : ALTER, CREATE, DROP ;
- un langage de contrôle des données et des utilisateurs (LCD) : GRANT, REVOKE.

Remarques : Beaucoup de systèmes de gestion de données (et non pas de gestion de bases de données) sont vendus comme étant relationnels, souvent parce qu'ils présentent les données sous forme de tables.

Un système est dit minimalement relationnel s'il satisfait aux conditions suivantes :

- toute information dans la base est représentée par des valeurs dans des tables,
- il n'y a pas de pointeurs visibles par l'utilisateur entre les tables,
- le système doit supporter au moins les opérateurs relationnels de restriction, projection, jointure naturelle.

Un système est dit complètement relationnel s'il satisfait, en plus, aux conditions suivantes :

- il supporte tous les opérateurs de l'algèbre relationnelle,
- il supporte la contrainte d'unicité de clé d'une relation,
- il supporte les contraintes référentielles qui permettent de s'assurer que la valeur d'une donnée d'une relation existe dans une autre relation (notion de **foreign key**).

En dépit de sa simplicité et de son élégance le modèle relationnel n'apporte pas une réponse satisfaisante à tous les problèmes des applications. Il faut :

- Pouvoir prendre en compte des "objets" structurés ainsi que les opérations qui leur sont associées (bases de données orientées objet).
- Prendre en compte des données peu structurées : textes, sons, images, graphiques (bases de données multi-média).
- Faire le pont avec l'intelligence artificielle afin de pouvoir déduire de nouvelles données à partir de celles existant déjà (bases de données déductives).

Conclusion

Les bases de données sont incontournables dans le domaine de l’informatique. Elles sont le fruit de l’évolution technologique et résolvent les problèmes liés à l’utilisation des systèmes de fichiers. Toutefois, pour être performantes, elles méritent d’être gérées avec des objectifs précis. Ces objectifs ont donc été assignés à des Systèmes de Gestion de Bases de Données (SGBD) afin de résoudre les problèmes créés par l’approche classique.