Chapitre 1 : introduction aux bases de données

1. Les bases de données

Avant l'arrivée des bases de données, les systèmes traditionnels étaient centrés sur les fonctions et les traitements et non pas sur les données. De ce fait, chaque application a la responsabilité de stocker et gérer ses données dans des fichiers. Les données dans les systèmes traditionnelles ont les particularités suivantes :

- la structuration et la description des données d'un fichier sont définies dans le programme de l'application ce qui fait qu'il n' y a aucune indépendance entre programme et données.
- Une même information peut être trouvée dans plusieurs fichiers.
- Une même information peut être présentée sur différent formats dans plusieurs fichiers (Description multiple des structures de fichiers)
- Conflit si accès simultané au fichier.

Exemple :

Produit	Quantité	Fournisseur	Adresse
Pinces	60	QUIN K	Grande rue
Plaques	500	L. VIS	Quais des brumes
Boulons	2000	L. VIS	Quais des brumes

Fig.1.1 : Exemple illustrant un fichier de données dans les systèmes traditionnels

Nous constatons dans l'exemple de la figure 1.1 que lors des mises à jour de cette table, nous pouvons tomber sur des incohérences. Par exemple :

- Si on souhaite mettre à jour l'adresse d'un fournisseur, il faut le faire partout ;
- Pour ajouter un fournisseur nouveau, il faut obligatoirement fournir des valeurs de produit et de quantité ;
- La suppression de Plaques ou Pinces fait perdre les informations concernant les fournisseurs de ces produits.

L'utilisation des bases de données (BD) est la nouvelle approche qui a vu le jour au début des années 60 et dont le but est de remédier aux problèmes des systèmes traditionnels.

Selon Thierry Lecroq [3], l'approche BD correspond à une triple évolution :

- Evolution des entreprises (volumes importants de données, centralisées ou réparties)

- Evolution des technologies (accroissement des performances, intégration des composants, diminution des coûts) ;
- Evolution des systèmes d'exploitation et des architectures (extension logicielle du matériel initial, architecture client-serveur et réseaux combinant de façon transparente des machines et des applications hétérogènes).

Il définit les bases de données comme étant un ensemble structuré de données enregistrées avec **le minimum de redondance** pour satisfaire simultanément plusieurs utilisateurs de façon sélective en un temps opportun.

Une autre définition des BD est celui de Chris L. Date [2] :
Une base de données est une collection de données **persistantes** et **pertinentes** utilisées par les applications de certaines **organisations.**
Dans cette définition l'auteur évoque deux caractères intrinsèques aux bases de données qui sont :

- La persistance : les données sont stables. C'est-à-dire elles diffèrent de toute donnée de nature transitoire (les données plus éphémère).
- La Pertinente : données ciblées et bien choisies, selon le besoin de l'application.

Les utilisateurs des bases de données peuvent être un simple individu avec une simple base de données privés (médecin, avocat, Etc.) ou une société complète avec une base de données partagée de volume très important (une banque, une université, .Etc.).

Caractéristiques des bases de données

La première caractéristique des bases de données est l'indépendance entre les données et l'application et cela dans différents points [3] :

- La structuration et la description des données dans les BD sont unifiées et séparées des programmes d'application.
- L'utilisateur de la base de données n'a pas à connaitre l'organisation physique des données.
- La gestion des données (stockage, modification, recherche) qui est étroitement dépendante de leur structuration est fournie par le **système de gestion de la base de données (SGBD)** ; Les applications communiquent avec les données à travers une interface de ce système (plus de détails sur les SGBD sont donnés dans la section suivante).

Avantage des bases de données par rapport aux systèmes traditionnels

Les avantages des BD par rapport aux systèmes traditionnels sont divers. Entre autre nous pouvons citer :

- La compacité : éliminer l'utilisation de grands volumes de fichiers pour le stockage mal structuré des données.
- La rapidité : recherche et mise à jour rapide d'information
- Intégrité des données : éliminer totalement ou partiellement la redondance dans les bases de données ce qui permettra d'éliminer l'incohérence lors d'une mise à jour par exemple (c'est le cas contraire dans les systèmes traditionnels).

- Partage des données : les différents utilisateurs de la base de données peuvent accéder aux mêmes données simultanément.

2. Les systèmes de gestion des bases de données

Un système de gestion de base de données est un logiciel qui permet aux utilisateurs (individu ou application) de communiquer avec la base de données. Il offre la possibilité de manipuler des représentations abstraites des données indépendamment de leur organisation et de leur implantation sur les supports physiques (mémoires). Un SGBD permet à l'utilisateur de décrire précisément **ce qu'il veut obtenir** et **non comment l'obtenir** [3].

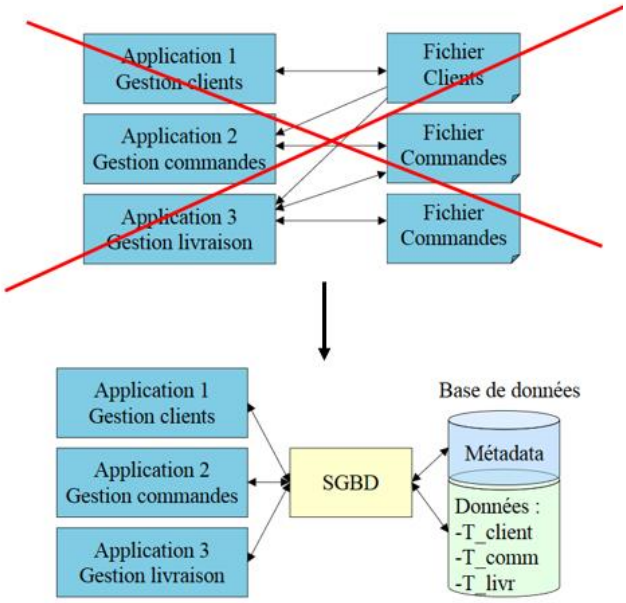


Fig. 1.2 : exemple illustrant le passage des systèmes traditionnels vers les bases de données [1].

D'une manière générale, ce logiciel se charge de gérer les différents aspects complexes (ou défis, par exemple : le partage, la restauration, .Etc.) liés aux donnée qu'un simple utilisateur de la base de données ne peux pas gérer. Parmi ces aspects nous citons [4] :

- **L'indépendance physique et logique des données**
  L'indépendance physique signifie l'indépendance entre les détails physique de stockage et d'accès aux données et les applications qui utilisent ces données. Dans les systèmes traditionnels, cette indépendance n'existait pas. Par exemple, un programmeur d'une application, pour manipuler des données stockées dans un fichier, doit connaitre l'organisation physique de ces fichiers et construira ses programmes en fonction de cette organisation. Un SGBD qui a la caractéristique d'indépendance physique, permet à l'administrateur de la base de données de modifier l'organisation physique (par exemple ajouter une table d'indexe) sans modifier les programmes utilisateurs.
  L'indépendance logique permet de modifier la conception des données (ajouter de nouvelles classes, nous nouvelles associations .Etc.) sans modifier les programmes utilisateurs.
- **Définition des données** :
  Le SGBD, avec un langage spécifique dit langage de définition des données (LDD), permet de définir les données sous forme non compilée et de les convertir dans la forme objet appropriée.

- **Manipulation des données** :
  Le SGBD, avec un langage dit langage de manipulation des données (LMD) permet de traiter les données par des requêtes d'interrogation (insertion, suppression, modification, .Etc.).
- **Efficacité des accès aux données :**
  Les SGBD visent à garantir un accès plus efficace que celui des systèmes de gestion des fichiers, et cela grâce au développement d'indexe sophistiqué, l'existence de plusieurs chemins d'accès à une donnée et l'existence de techniques d'optimisation de requêtes qui sélectionnent le chemin optimal à une donnée.
- **Administration centralisée :**
  Le SGBD donne un accès particulier à un utilisateur dit « administrateur de la base de données » pour administrer les données. La tâche d'administration confiée à cet utilisateur consiste à définir les structures de stockage et les structures de données ainsi qu'assurer le contrôle et le suivi de leur évolution.
- **Non redondance :**
  La conception des données à l'aide des modèles de données (par exemple le modèle relationnel) permet d'éviter les redondances.
- **La cohérence :**
  La vérification de la cohérence par le SGBD consiste à vérifier, à chaque fois que des données sont sollicitées, la satisfaction d'un ensemble de règles appelés contraintes d'intégrité (CI) définies sur ces données. Nous dirons qu'une **BD est cohérente si les CI sont toujours vérifiées lors d'opérations sur cette base.**
- **La partageabilité :**
  Le SGBD doit permettre l'accès concurrent aux données (partage des données entre plusieurs applications). Cela signifie qu'il doit pouvoir détecter les situations conflictuelles et d'y remédier. Pour la gestion des accès concurrent, le SGBD utilise la notion de **Transaction** et **la définition d'algorithmes de gestion de la concurrence.**

  Une transaction est un ensemble indivisible d'opérations sur la BD dont l'exécution maintient la BD dans un état cohérent.
  Pour dire qu'un ensemble d'opérations constitue une transaction, quatre propriétés doivent être satisfaites, elles sont appelées les propriétés **ACID** [5].
  A : c'est la propriété d'**Atomicité** qui consiste à interdire l'exécution partielle de la transaction. Quand une transaction en cours d'exécution est interrompue, le SGBD doit annuler toutes les opérations déjà exécutées.
  C : c'est la propriété de la **Cohérence** qui consiste à assurer la cohérence des données. La transaction doit respecter les CI pour donner un résultat cohérent (passer d'un état valide à un autre état valide).
  I : c'est la propriété d'**Isolation** qui consiste à exécuter chaque transaction en isolation des autres transactions ; les modifications qu'apporte une transaction à la base de données durant son exécution ne seront visibles par les autres transactions qu'à sa terminaison. Cette propriété assure la cohérence des données dans le cas de l'accès concurrent entre les transactions. Pour garantir cette propriété, le SGBD adopte toute une politique basée sur le principe de verrouillage/ déverrouillage des ressources.
  D : c'est la propriété **Durabilité** qui signifie que toutes les actions entreprises par les transactions sur les données doivent avoir le caractère durable ; on ne peut pas refaire une transaction terminée. Pour cela, le SGBD doit enregistrer toutes les chronologies des

interactions et modifications effectuées sur les données (dans des fichiers d'historique dites journaux). Le but est de se prémunir contre les éventuelles pannes informatiques.

- **Sécurité et confidentialité :**
  Le SGBD est responsable de protéger la base de données contre les pannes et les accès mal intentionnés.

  Protection contre les pannes
  Il peut s'agir de pannes simples comme la perte des données de la mémoire centrale ou des pannes plus graves caractérisées par la perte des données de la mémoire secondaire.
  Comme déjà vu dans la description du **D** des propriétés ACID, le SGBD assure la gestion des journaux de transaction afin de garder la trace d'exécutions antérieures ce qui permet de récupérer un état cohérent de la base de données en cas de pannes catastrophiques.

  Protection contre les pannes mal intentionnées
  Le SGBD doit gérer la confidentialité des données en utilisant par exemple les mots de passe et la définition des droits d'accès et des privilèges.

3. Les niveaux de représentation des données

L'un des objectifs primordiaux des SGBD est de séparer les manipulations faites par les utilisateurs sur la base de données de la description conceptuelle (indépendance logique) et de l'organisation physique (indépendance physique) de cette dernière. Dès la fin des années 60, la solution pour assurer cette indépendance fut proposée : la notion du schéma à trois niveaux du groupe ANSI-SPARC, (l'architecture **ANSI-SPARC**) sur laquelle reposent pratiquement tous les SGBD aujourd'hui [5] (figure 1.3). L'objectif de cette architecture est de donner une souplesse permettant l'évolution des applications au moindre coût.

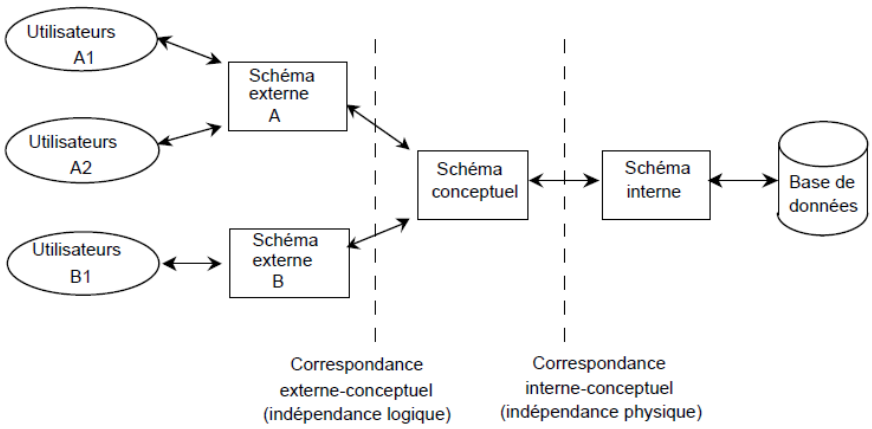


Fig. 1.3 : différent niveau de représentation des données [8]

L'architecture **ANSI-SPARC** est divisée en trois niveaux, celui du schéma interne (SI), celui du schéma conceptuel (SC) et celui des schémas externes (SE). Un schéma permet correspond à une représentation particulière des données.

Le schéma conceptuel

Ce schéma est le résultat de la phase de conception dans le processus de développement d'une base de données. Il consiste à représenter les objets du monde réel en objets issus des **modèles** bien définis (nous allons détailler ces modèles dans la section suivante).

Le schéma interne

Ce schéma est le résultat de la phase de réalisation dans le processus de développement d'une base de données. Il représente l'organisation des données dans les supports de stockage physique. Les détails de cette implémentation sont définis par le SGBD choisi. Une base de données est définie et manipulée à travers le schéma conceptuel sans se soucier des détails du schéma interne.

Le schéma externe

Alors que les schémas conceptuel et interne décrivent toute la base de données, le schéma externe appelé aussi **VUE** permet de représenter une partie de la base de données. Cette représentation correspond à une perception des données par un utilisateur. Ce dernier ne peut manipuler que les données appartenant à son propre schéma externe. Le schéma externe est un **sous-schéma** d'un schéma conceptuel.

Les différentes vues permettent la validation du schéma conceptuel (il existe une technique de construction du schéma conceptuel à partir de l'intégration des vues) [4].

4. Les modèles de données

Un modèle de données est le **plan de conception** sur lequel se base SGBD. Il donne des détails sur comment structurer les données, les relations entre eux, les règles et les contraintes de validation des données et toutes les transformations pouvant être appliquées sur les données [7]. Dans l'architecture ANSI-SPARC, le schéma conceptuel constitue **une instance** d'un modèle de données.

Depuis l'arrivée des bases de données, différents modèles ont été proposés. Nous pouvons les classer en trois catégories liées aux différentes générations des bases de données [4]. La première génération (années 60) englobe les modèles hiérarchiques et les modèles réseau. La deuxième génération (années 70) englobe des modèles plus évolués à l'instar des modèles Entité/Association, relationnel et réseau sémantique. La troisième génération (années 90) a vu l'émergence des modèles pour des données plus complexes, ce sont les modèles objet.

4.1. Le modèle hiérarchique

Le premier SGBD basé sur le modèle hiérarchique est le système IMS (Information Management System) développé par IBM en 1968. Les modèles hiérarchiques se caractérisent par **une forte dépendance entre la représentation logique et physique** des données. Ainsi, ils ne sont pas conforment à l'architecture ANSI-SPARC.

Un modèle hiérarchique utilise une structure d'arbre pour la représentation des données. Cette structure est constituée de la racine liée à plusieurs niveaux de sous arbre. Un nœud parent de l'arbre peut être lié à un ou plusieurs nœuds fils (nœud de niveau inférieur). Un nœud fils a un seul nœud

parent. Dans ce modèle, nous appelons un type **enregistrement** un ensemble de noms d'attributs avec leurs types correspondants. Selon la terminologie du système IMS, un nœud est appelé **segment** et il correspond à une instance d'enregistrement. L'unité d'accès est le segment et l'on ne peut pas accéder à un segment sans accéder à son supérieur hiérarchique. Aussi, la suppression d'un segment entraine celle de tous les segments subordonnés.

Prenant l'exemple de gestion des approvisionnements décrit dans le livre de Date [2] et repris dans le cours de Mokhtari [4].

- Un fournisseur est décrit par son numéro NF, son nom NOM, son code CODE et la ville où il est situé VILLE.
- Une pièce est décrite par son numéro N, son nom NOM, son poids POIDS, son matériau MATERIAU et la ville où elle est stockée VILLE.
- Un fournisseur f fournit une pièce p en une certaine quantité q.
- Un fournisseur peut fournir plusieurs pièces et une pièce peut être fournit par plusieurs fournisseurs (association n-m).

Nous voulons concevoir cette partie de la base de données qui décrit la relation entre les entités pièce et fournisseur en utilisant le modèle hiérarchique. Une solution de conception de cette relation est celle donnée par le schéma la figure 1.4.

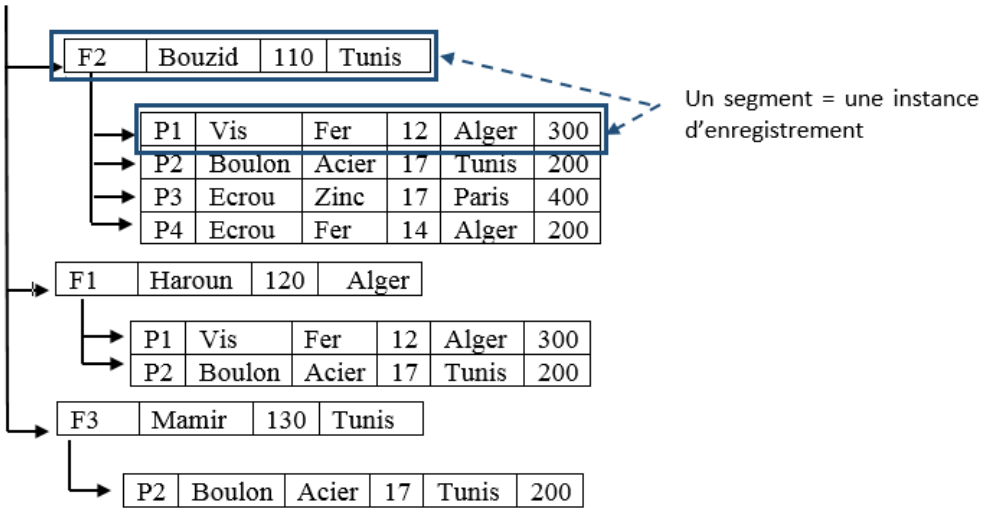


Fig. 1.4 exemple de la structure hiérarchique

La première difficulté est que **ce modèle ne permet pas de représenté les associations plusieurs à plusieurs**. Par conséquent, le concepteur doit choisir de représenter la relation entre pièce et fournisseur en un seul sens (association un à plusieurs : un fournisseur fournit un ou plusieurs pièce).

La deuxième difficulté est la dépendance **entre la structure hiérarchique et la formulation des chemins d'accès** (pour accéder à un segment il faut accéder d'abord à son segment supérieur). Par conséquent, les opérations de recherche et de mise à jour deviennent plus complexes, **d'autant plus avec un mauvais schéma conceptuel.** Revenant à l'exemple de la figure 1.4.

Dans le cas de la recherche : si l'on veut établir la liste des pièces, il faudra analyser **pour chaque fournisseur** la liste des pièces fournies : donc on doit passer obligatoirement par le segment fournisseur avant d'accéder aux segments pièces.

Dans le cas de l'insertion : si l'on veut insérer une nouvelle pièce non encore fournie par aucun fournisseur, on doit introduire un fournisseur fictif.

Dans le cas de la suppression : si l'on supprime le seul fournisseur d'une pièce donnée, les informations liées à cette pièce seront aussi supprimées.

Dans le cas de la modification : Si l'on veut modifier le matériau de P2, on doit examiner toutes les listes des pièces fournies par les fournisseurs (analyse complète de la base de données).

La troisième difficulté est dans les requêtes d'interrogation de la base de données qui nécessitent que l'utilisateur du SGBD y une haute compétence technique pour les formuler. La formulation des requêtes dépend de la structure hiérarchique ce qui rend la mise à jour des programmes d'application difficile.

4.2. Le modèle réseaux

Le premier objectif de ce modèle est de régler le problème de dépendance entre les nœuds parent-fils rencontré dans le modèle hiérarchique. La première définition des modèles réseaux était en 1969 lors de la conférence CODASYL (Conference on Data Systems Languages). Dans ce modèle, nous appelons :

- **Atome** (data item) : la plus petite unité de données possédant un nom et représentée par une valeur
- **Agrégat** (data aggregate) : une suite d'atomes rangés consécutivement dans la base de données et possédant un nom.
- **Enregistrement** (record) : une suite d'atomes et d'agrégats rangés consécutivement dans la base de données et constitue le bloc d'échange entre les applications et la BD.

Par analogie, un nœud dans le modèle réseau correspond à un type d'enregistrement. Chaque enregistrement est identifié de manière unique par une clé (une clé est un atome ou un agrégat). Les arcs définissent les relations entre les enregistrements.

Dans le modèle de CODASYL un lien est appelé **SET** et il définit un lien entre un **enregistrement propriétaire appelé OWNER** et un ou plusieurs **enregistrements membres appelés MEMBER**. Un enregistrement ne peut pas être à la fois propriétaire et membre d'un même SET mais il peut être propriétaire de plusieurs SET différents et/ou membre de plusieurs SET différents. Le lien peut posséder des propriétés qui relient les enregistrements appelés les **données d'intersection**.

Dans la norme CODASYL, les liens sont de deux types uniquement 1-1 et 1-N. Pour créer un lien M-N on peut le faire indirectement : utiliser un **enregistrement intermédiaire** avec deux liens 1-N. l'enregistrement intermédiaire est créé en utilisant les données d'intersection si elles existent. Il possédant comme clé la concaténation des clés des enregistrements composant le lien (le lien avant transformation ; le lien M-N).

L'exemple précédant décrivant la relation fournisseur-pièce est représenté par le modèle réseau CODASYL de la figure 1.5.

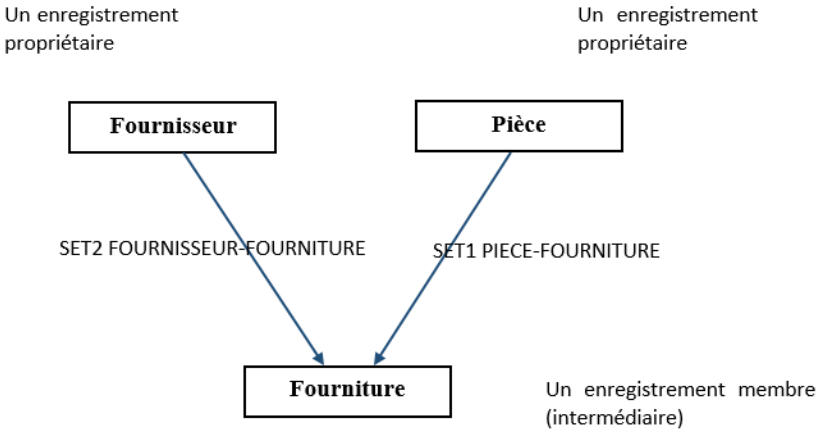


Fig. 1.4 exemple de la représentation réseau CODASYL

Pour définir la relation M-N entre fournisseur et pièce, d'abord on crée l'enregistrement intermédiaire « Fourniture » puis on crée deux liens (SETS de type 1-N) pour relier pièce et fournisseur à fourniture. Ce nouvel enregistrement possède comme clé la concaténation des clés des enregistrements composants le lien.

Le modèle réseau CODASYL est implémenté grâce à la liste circulaire ou anneau. Cette implémentation est présentée dans le schéma de la figure 1.5 appelé le schéma d'occurrences. L'exemple de la figure 1.5 est un exemple de base de données décrite précédemment.

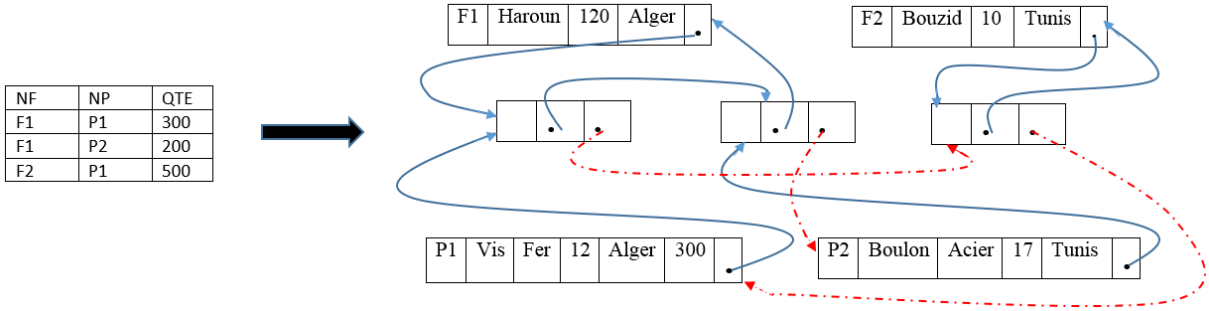


Fig. 1.5 exemple d'implémentation du modèle réseau CODASYL

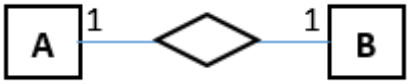
Avec une telle implémentation, les opérations d'interrogation de la base de données sont beaucoup plus simplifiées, par exemple on peut insérer une pièce sans qu'elle soit fournie par un fournisseur, on peut supprimer les fournisseurs sans supprimer des pièces et on peut faire des modifications sans problèmes particuliers.

Le majeur inconvénient du modèle réseau est la gestion complexe des pointeurs. Le parcours des chainages devient plus délicat à cause des interactions des instructions du LMD sur ces pointeurs.

4.3. Le modèle Entité / Association

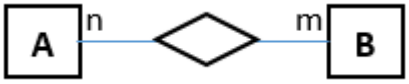
Le modèle Entité / Association décrit les données par un formalisme plus proche de la réalité. La base de données est représentée par un ensemble d'entités, leurs propriétés (attributs), les associations entre les entités et les contraintes auxquelles elles sont soumises. Une entité décrit un objet du monde réel. Les propriétés décrivent l'entité et parmi eux une ou plusieurs peuvent être des clés.

Graphiquement, les entités sont présentées par des rectangles et les associations par des losanges. Les cardinalités décrivent le nombre d'associations auxquelles une entité peut participer. Il existe trios cardinalités prédéfinies :



à une instance de l'objet A est associée une et une seul instance de l'objet B et inversement.

à une instance de l'objet A est associée une et une seule instance de l'objet B et à une instance de l'objet B est associée une ou plusieurs instances de l'objet A.



à une instance de l'objet A est associée une ou plusieurs instances de l'objet B et inversement.

Le type en intension et en extension

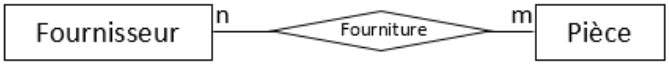
Les entités, propriétés et associations sont classées et définies par des **TYPES**. Un type définit une population d'objets de la même nature. On parle aussi de classe d'entité, classe d'association ou classe de propriété. Le Type peut être défini en intension ou en extension.

Exemple :

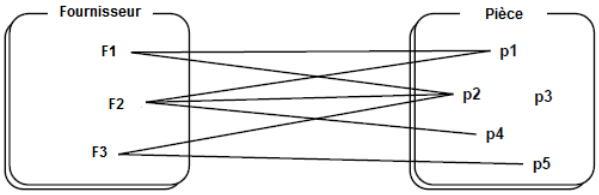
Basé sur l'exemple précédent, nous appelons :
L'entité type « Fournisseur », l'association type « Fourniture » (association entre pièce et fourniture)

1) L'entité type « Fournisseur » en intension est : Fournisseur (NF, Nom, CODE, Ville)
L'entité type « Fournisseur » en extension (instance de la classe entité « fournisseur ») est :
{F1, Haroun, 100, Alger}
{F2, Ali, 220, Tunis}

2) L'association type « Fourniture » en intension est le schéma de l'association type suivant :



L'association type « Fourniture » en extension (on dit aussi l'extension de la classe d'association « Fourniture ») est :



Rappelons qu'une clé est un attribut ou un ensemble d'attributs dont les valeurs permettent d'identifier de manière unique les entités d'un même type. Cela implique une contrainte d'unicité sur l'extension de l'entité type. La clé d'une association type est la combinaison des clés des entités

types participantes (par exemple dans l'association type « Fourniture », la clé est une combinaison des clés NF et NP).

Le modèle Entité/Association n'est pas utilisé comme étant un modèle à part entière mais beaucoup plus comme un outil d'aide à la conception des schémas relationnels que nous verrons en détails dans la section suivante. De plus, il existe un processus simple qui permet le passage du modèle Entité/ Association à un modèle relationnel.

4.4. Le modèle relationnel

Le modèle relationnel est la tendance des bases de données actuelles. Il est basé sur des principes mathématiques ce qui facilite sa compréhension et il est indépendant des détails d'implémentation.

Le modèle relationnel se base sur les trois aspects suivants :

- L'aspect structural : les données sont représentées uniquement par des tables appelées relations. Dans cette structure on trouve un ensemble de concepts comme attributs, tuple, clé primaire, Etc.
- L'aspect d'intégrité : les tables doivent respecter un ensemble de contraintes d'intégrité.
- L'aspect de manipulation : le principe des opérations qui manipulent les tables est de générer de nouvelles tables à partir des tables existantes.

La figure 1.6 montre un exemple de base de données représentée par le modèle relationnel.

FOURNISSEUR

NF	NOM	CODE	VILLE
F1	Haroun	120	Alger
F2	Bouzid	110	Tunis
F3	Kaci	130	Tunis

PIECE

NP	NOM	MATERIAU	POIDS	VILLE
P1	Vis	Fer	12	Alger
P2	Boulon	Acier	17	Tunis
P3	Ecrou	Zinc	17	Paris
P4	Ecrou	Fer	14	Alger

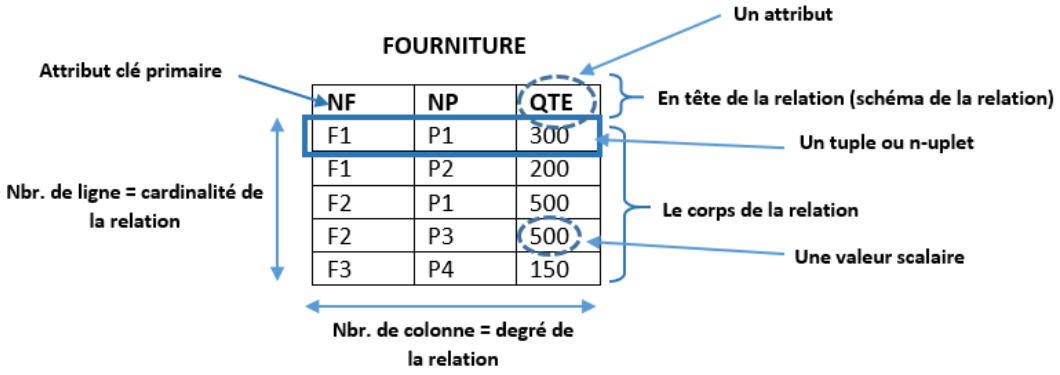


Fig. 1.6 exemple de la représentation relationnelle

4.4.1. L'aspect structural

Dans l'exemple précédent nous avons illustré les concepts de base du modèle relationnel.

La valeur atomique dans le modèle relationnel est appelée **valeur scalaire**, elle représente la plus petite unité sémantique de données telle que : F1, Alger, Haroun. Etc.

Un **attribut** définit une caractéristique d'une entité dans le monde réel, comme par exemple l'attribut NF, l'attribut NOM, l'attribut Quantité. Etc.

Un **domaine** est l'ensemble des valeurs scalaires toutes de même type qui définissent les valeurs que peut prendre un attribut. Par exemple le domaine des villes, le domaine des numéros de fournisseurs. Etc.

Un attribut est défini sur un seul domaine : les valeurs de l'attribut sont tirées de ce domaine. Par exemple l'attribut NF dans la table Fournisseur et dans la table Fourniture est défini dans le domaine des numéros de fournisseur. Les valeurs de NF sont des sous-ensembles des valeurs du domaine sur lequel cet attribut est défini.

Les attributs dans une relation doivent avoir des noms différents mais ils peuvent être tirés d'un même domaine.

Un **tuple** est une suite de valeurs d'attributs à un instant donné. Les tuples existent dans la table d'une manière désordonnée et sans répétition (on ne peut pas trouver le même tuple dupliqué).

Une **relation R** est constituée de deux parties : l'entête et le corps.

L'ensemble des tuple constitue le corps de la table (ensemble de lignes). L'entête de la table ou le schéma de la relation est défini par un ensemble fixé d'attributs.

Le nombre de n-uplet dans la table définit **la cardinalité** de la relation et le nombre d'attributs définit **le degré** de la relation.

Ainsi, nous pouvons dire qu'une relation correspond à une table, un n-uplet correspond à une ligne et un attribut à une colonne.

Les tuples dans une table sont identifiés par des **clés** qui sont de trois types :

- Clé candidate : une clé candidate est un ensemble d'un ou plusieurs attributs de la relation qui permettent d'identifier d'une manière unique un tuple. Date [2] donne la définition suivante pour une clé candidate :
  *Soit R une relation avec les attributs A_1, A_2, … A_n, l'ensemble C = (A_i, A_j, …A_k) constitue une clé candidate de R si et seulement s'il satisfait les conditions d'**unicité** et d'**irréductibilité**.*
  L'unicité veut dire qu'à un instant donné il ne doit pas exister deux tuples dans R qui possèdent la même valeur de A_i, la même valeur de A_j, …. et la même valeur de A_k
  L'irréductibilité veut dire qu'on ne peut pas réduire l'ensemble C sans perdre la propriété d'unicité.
  Une relation peut avoir plusieurs clés candidates et au moins elle possède une clé candidate qui est la combinaison de tous les attributs de la relation (dans le cas où l'ensemble satisfait les conditions d'unicité et l'irréductibilité).
- Clé primaire : une clé primaire est la clé choisie depuis les clés candidates pour identifier les tuples. Si la relation a plus d'une clé ; les autres clés non choisies sont appelées **clés alternatives**.
- Clé étrangère : soit R2 une relation de la base de données, alors une clé étrangère CE dans R2 est un attribut (ou une combinaison d'attributs) de R2 dont **les valeurs doivent correspondre** à celles de la clé primaire d'une autre relation R1.

4.4.2. L'aspect d'intégrité

Afin de préserver la cohérence des données, tout schéma conceptuel doit respecter un ensemble de contraintes d'intégrité. Ces contraintes sont un ensemble de règles qui doivent être respectées à

chaque exécution d'opération sur la base de données. Dans le cas contraire l'opération doit être rejetée. Nous décrivons ces contraintes dans les points suivants :

- Contrainte d'intégrité d'entité : les attributs qui sont des clés primaires ne peuvent pas avoir la valeur null (null veut dire une valeur indéfinie ou inexistante).
- Contrainte d'intégrité référentielle : si une relation R2 comprend une clé étrangère CE correspondant à la clé primaire C d'une autre relation R1, chaque valeur de CE dans R2 doit être égale à la valeur de C dans un tuple de R1 ou **être totalement nulle (chaque valeur d'attribut participant à cette valeur CE doit être null).**
- Contrainte d'intégrité sémantique : les règles que les instances des relations doivent satisfaire pour que les données restent conformes à la réalité qu'elles représentent. Ces contraintes sont ajoutées par l'administrateur de la base de données et vérifiées par le SGBD. Un SGBD relationnel propose plusieurs types de contraintes sémantiques entre autres : la contrainte de domaine (les valeurs d'un attribut doivent appartenir aux sous-ensembles des valeurs du domaine sur lequel cet attribut est défini), la contrainte NOT NULL (un attribut ne doit pas avoir la valeur null), la contrainte d'unicité (les valeurs d'un attribut doivent être différentes) et la contrainte individuelle (une contrainte que doit vérifier individuellement par exemple l'année de fabrication d'un véhicule doit être inférieur à l'année actuel du véhicule).

4.4.3. L'aspect de manipulation

La manipulation de la base de données afin d'en extraire les informations souhaitées est une tâche très importante de tout SGBD. Ainsi, un langage d'interrogation de la base de données doit être défini par le SGBD au profit de l'utilisateur. Dans le cadre de ce module, nous allons voir deux types de langages d'interrogation de base de données qui sont le langage algébrique et le langage SQL.

Références bibliographique

[1] Blancheflour Dumont, « Systèmes de Gestion de Bases de Données (Relationnelles) ». Site : https://slideplayer.fr/slide/1301574/

[2] Chris Date. Introduction to data base systems. livre, 1995.

[3] Thierry Lecroq, « les bases de données ». Cours de l'université de de Rouen.

[4] Aicha Mokhtari, « introduction aux bases de données et aux systèmes de gestion de base de données ». Cours de l'USTHB.

[5] Jacques Printz, « Architecture des logiciels ». Livre 2006.

[6] « Cours complet pour apprendre les systèmes de gestion de bases de données ». Developpez.com. site : https://sgbd.developpez.com/tutoriels/cours-complet-bases-de-donnees

[7] Neeraj et al. "Data fundamentals". Livre 2010.

[8] J. Darmont. "Base de données ". Cours du Centre associé de Clermont-Ferrand 1998.