Généralités sur les bases de données :

Les systèmes de gestion de bases de données

Plan du module

I. RAPPELS SUR LES FICHIERS 4

A. RAPPELS SUR LES FONCTIONS DU SYSTEME D'EXPLOITATION 4
B. LA STRUCTURE LOGIQUE DES FICHIERS 5
1. Structure logique de longueur fixe 5
2. Structure logique de longueur variable 7
C. LA STRUCTURE PHYSIQUE DES FICHIERS 7
1. Enregistrement logique, enregistrement physique 7
2. Structure physique des fichiers 8
D. LES ORGANISATIONS ET LES ACCES 9
1. Les organisations 9
2. Les accès. 11
Organisations et accès 12

II. LES SYSTEMES DE GESTION DE FICHIERS 13

A. GENERALITES 13
1. Fonctionnement d'un SGF 13
2. Objectifs d'un SGF 13
B. LES INCONVENIENTS DES SGF 14
1. Exemple : Gestion des stagiaires de l'ESAT 14
2. Inconvénients directs 14
3. Inconvénients indirects 15

III. NOTIONS DE SGBD 17

A. DEFINITIONS 17
1. Base de données 17
2. Le Système de Gestion de Bases de Données 17
B. LES OBJECTIFS D'UN SGBD 18
1. Centraliser l'information 18
2. Assurer l'indépendance données - traitements 18
3. Permettre les liaisons entre ensembles de données 18
4. Intégrité et cohérence 18
5. Partage des données 19
6. Sécurité 20
7. Confidentialité 22

IV. LES SGBD DE 1ERE GENERATION 23

A. PRINCIPES 23
B. LES MODELES HIERARCHIQUES 23
1. Innovation 23
2. Concepts associés 25
3. Avantages 26
4. Inconvénients 26
C. LE MODELE RESEAU (DIT NORME CODASYL) 27
1. Innovation 27
2. Présentation 27
3. Avantages 29
4. Inconvénients 29
5. Passage du modèle conceptuel des données au modèle logique "CODASYL" 29
6. Application 31

V. LES SGBD RELATIONNELS 32

VI. LES SGBD OBJETS ET LES AUTRES FORMES DE SGBD 32

A. LES SGBD OBJETS 32
B. LES NOUVELLES FORMES DE SGBD 32
1. Les bases de données réparties 32
2. Les info-centres 33

VII. LES ACTEURS D'UN SGBD 33

A. LES ADMINISTRATEURS 33
1. L'administrateur système (ou root ou administrator) 33
2. L'administrateur du SGBD (ou sa : system administrator) 33
3. L'administrateur de base de données (ou dba : data base administrator) 33
4. L'administrateur des données 34
B. LES UTILISATEURS" INFORMATICIENS" 34
1. Les programmeurs d'applications 34
2. Les utilisateurs occasionnels 34
C. LES UTILISATEURS NON INFORMATICIENS (OU EXPLOITANTS) 34


La gestion des informations se faisait jadis à partir de fiches manuscrites puis dactylographiées, classées bien souvent selon un index

Exemple : Cas d'une gestion de personnels par ordre alphabétique sur le nom, avec un index ou coche sur la première lettre du nom, pour améliorer le classement et la recherche.


Ces fiches regroupaient des informations se reportant à un même sujet.

Exemple : Selon l'exemple précédent : le nom, le prénom, la date de naissance, l'adresse, les qualifications, l'emploi tenu, l'indice de rémunération ...), avec éventuellement une information particulière qui servait à identifier plusieurs personnels ayant le même nom et le même prénom, soit un code ou un numéro matricule.


Ces principes de regroupement d'informations, d'identifiant, de fiche et de fichier servirent de modèle aux premiers fichiers informatiques : notion qui apparut dans les années cinquante.


Avec l'informatique, un certain nombre d'opérations ont été automatisées et l'enregistrement des informations est donc passé de la fiche manuelle à un support informatique (bande magnétique, disque dur, disquette etc...). Ces informations ont été organisées dans des fichiers.

La problématique générale est alors la suivante : comment, à partir d'un simple ordinateur, réussir à gérer toutes les informations d'une entreprise ?

Les connaissances en architecture système, en analyse et en programmation ne suffisent pas à elles seules à résoudre le problème du stockage des données et de leur traitement. Mais l'on peut, pour commencer, se servir des structures et des mécanismes propres à la gestion des fichiers pour organiser le stockage des données.

I. Rappels sur les fichiers

A. Rappels sur les fonctions du Système d'Exploitation

Tout Système d'Exploitation dispose d'un Système de gestion des fichiers, responsable de l'organisation et de l'accès aux informations.

> ***Fichier* : Ensemble organisé de regroupements d'informations de même nature, appelés articles ou enregistrements, susceptibles de faire l'objet de traitements par des programmes, ou produits par ces mêmes programmes.**

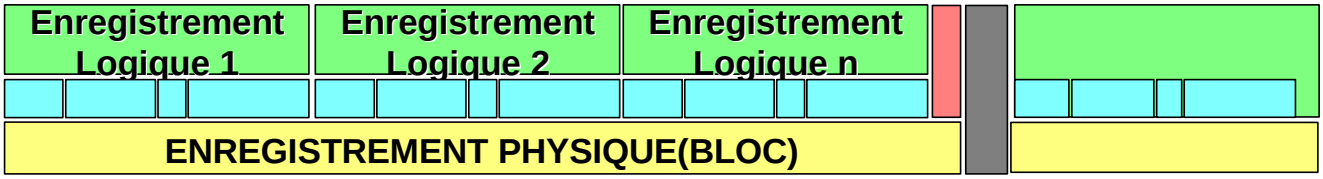
Parmi les fichiers que gère un système d'exploitation, on peut distinguer : les fichiers de données, les fichiers programmes ou les fichiers textes.

On ne s'intéresse ici qu'aux fichiers de données.

Ces fichiers de données sont destinés à faire l'objet de traitements par des programmes dans le but de création, de modification, de suppression et/ou de consultation des informations qu'ils contiennent.

En conséquence le SE considère un **fichier** comme une unité logique qu'il doit pouvoir gérer en fonction des espaces physiques disponibles sur les supports périphériques alors que l'utilisateur le considère comme une identité de dimension arbitraire qui forme un tout.

La notion de fichier physique (*contenant*) s'oppose donc à la notion de fichier logique (*contenu*) et il faut bien comprendre ces deux notions pour savoir gérer les informations.

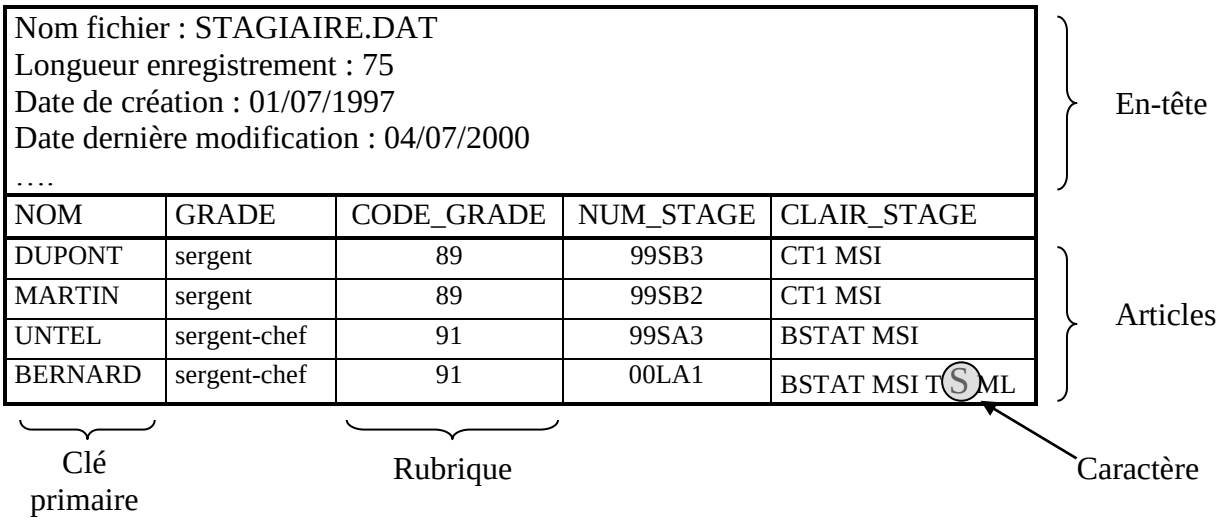


B. La structure logique des fichiers

> ***Structure logique** : Ordre de consultation des enregistrements de façon à n'obtenir que des enregistrements d'un même fichier*

1. Structure logique de longueur fixe

Avant même de commencer à stocker des informations, on doit créer une structure pour le fichier :



> ***En-tête (ou header)**: ensemble de données décrivant un fichier avec le détail de ses caractéristiques (organisation, longueur des enregistrements, date de création, date de dernière modification etc…).*
>
> ***Enregistrement logique (ou Article):** unité de base directement accessible par le programme utilisateur ; Ensemble structuré de rubriques se rapportant à un même sujet.*

Exemple : L'article STAGIAIRE comporte les rubriques NUM_STAGE, CLAIR_STAGE…

Chaque enregistrement se voit attribuer un numéro d'ordre (numéro ou clé).

> ***Rubrique (attribut ou champ ou donnée (datum)) : le nom attribué à chaque zone contenant une valeur en clair ou codifiée.***

Exemple : La rubrique GRADE contenant la valeur "sergent" et la rubrique CODE_GRADE contenant la valeur 89.

> ***Caractère** : le plus petit élément de donnée dans une rubrique. On peut dire que c'est le plus petit groupement de bits (BInary digiT) contigus et significatifs.*
>
> ***Clé** : est constituée d'une (ou de plusieurs) donnée(s) qui permet(tent) le rangement et la localisation de l'article dans le fichier de manière optimisée. La clé appartient généralement à l'article. Primaire, elle identifie l'article ; secondaire elle peut tolérer des valeurs multiples (doublons) et permet un accès plus rapide à tous les enregistrements pour lesquels la clé secondaire aurait la valeur recherchée.*

La clé primaire est toujours renseignée. Elle a un contenu unique pour chaque enregistrement logique, elle appartient à l'article en tant que rubrique, son contenu est ou non numérique. La clé primaire est utilisée par le SGF pour la construction des tables d'index.

1.1. Structure logique séquentielle classique

On accède à un enregistrement après avoir lu tous ceux qui le précèdent.

IDENT	**NOM**	**COEF**
1	DUPONT	30
2	MARTIN	23
3	UNTEL	28
4	BERNARD	12

Les enregistrements sont rangés dans l'ordre logique de lecture. Pour respecter cet ordre, on est souvent amené à trier les enregistrements, en les déplaçant. Ce type de tri est long et fastidieux ; le risque d'erreur important. Le tri ne peut se faire que sur un seul critère.

1.2. Structure logique séquentielle chaînée

Dans ce type de structure, les enregistrements peuvent être dans n'importe quel ordre.
Pour conserver un ordre logique des enregistrements, on ajoute une rubrique spéciale à la structure pour indiquer la position de l'enregistrement logique suivant ; cette rubrique contient l'adresse de cette enregistrement : c'est un « pointeur ».

IDENT	**NOM**	**COEF**	**Suivant**
1	UNTEL	28	@4
2 = début	DUPONT	30	@3
3	MARTIN	23	@1
4	BERNARD	12	Fin

Cette fois-ci, pour effectuer des tris, il suffit simplement de reconstruire le "chaînage" entre les différents articles à partir des "pointeurs", ce qui est plus rapide et plus sûr.
Cette structure a aussi l'avantage de proposer la notion de ***suppression logique*** des articles (l'enregistrement est présent dans le fichier, mais aucun pointeur ne permet d'y accéder).

1.3. Structure logique indexée

Pour accéder encore plus vite aux enregistrements, on ajoute au fichier de données un fichier index, trié en fonction de la rubrique la plus intéressante (la clé primaire par exemple).

Ce fichier index ne comporte plus que le numéro d'enregistrement, la rubrique servant d'index et l'adresse de l'enregistrement logique suivant (selon le critère d'indexation).

IDENT	**NOM**	**COEF**
1	UNTEL	28
2	DUPONT	30
3	MARTIN	23
4	BERNARD	12

Fichier de données

N°	**NOM**	**Adresse**
1	BERNARD	@4
2	DUPONT	@2
3	MARTIN	@3
4	UNTEL	@1

Fichier d'index (sur le nom)

Le fichier de données ne contient que des données, le fichier d'index peut être généré à n'importe quel instant, les accès se font par l'intermédiaire du fichier index, plus petit et plus maniable.

On peut, en outre, créer plusieurs fichiers index pour un même fichier de données (tri sur le nom, sur la date, sur un ensemble d'articles…).

2. Structure logique de longueur variable

Pour éviter le gaspillage d'espace on peut utiliser un fichier à structure variable :

- Les champs sont séparés par un séparateur spécifique : **\00** par exemple $00_{(16)}$;
- Les enregistrements sont séparés par un autre séparateur : **\n** par exemple $13_{(16)}$.

Le gain de place n'est pas négligeable mais la gestion devient plus complexe.

C. La structure physique des fichiers

Si l'on se penche maintenant sur la structure physique des fichiers (contenant), on se rend compte que des principes similaires à ceux énoncés précédemment sont mis en application, mais les mécanismes sont intimement liés aux possibilités de gestion des entrées-sorties du SE présent.

> ***Structure physique** : Ordre physique des enregistrements.*
>
> ***Bloc** : est constitué de un à plusieurs enregistrements logiques traités comme un tout au niveau des entrées-sorties.*

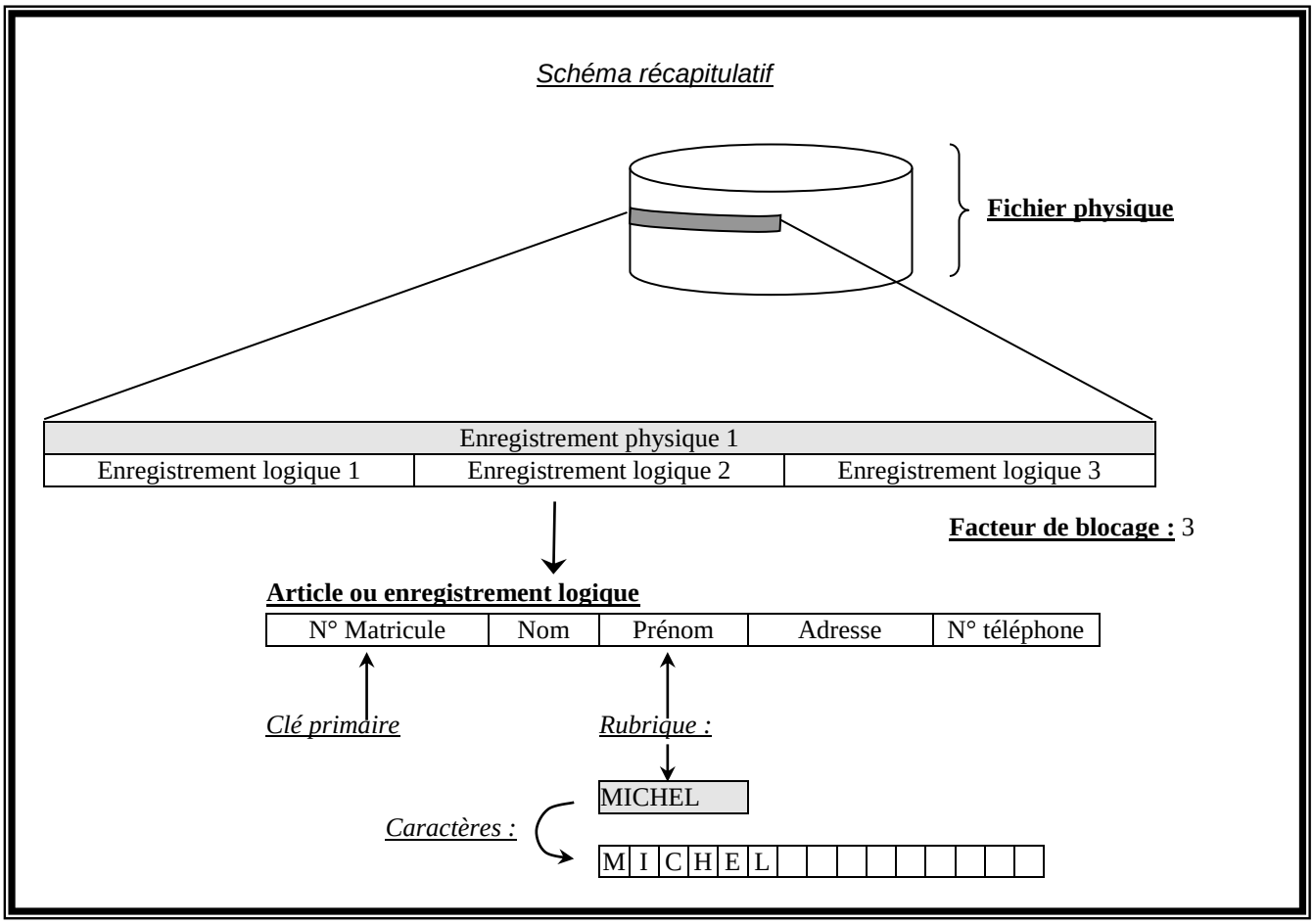
L'intérêt de ce principe réside, en tenant compte de la taille des buffers, dans la diminution du nombre d'entrées-sorties physiques.
De plus, les principes mis en place pour la gestion des mémoires secondaires autorisent le fractionnement d'un fichier physique en plusieurs endroits.

1. Enregistrement logique, enregistrement physique

> ***Enregistrement physique (ou bloc)** : ensemble des caractères transférés par le système d'exploitation au cours d'une entrée-sortie.*
>
> ***Enregistrement logique :** unité de base directement accessible par le programme utilisateur.*

Les tailles de ces différents enregistrements ne sont pas obligatoirement compatibles : le SE doit permettre de faire la correspondance entre physique et logique.

> ***Facteur de groupage (ou facteur de blocage) : Nombre d'enregistrements logiques contenus dans un enregistrement physique.***



2. Structure physique des fichiers

2.1. Structure physique séquentielle

Ceci est le cas des bandes magnétiques : les enregistrements sont écrits les uns à la suite des autres.
Il est donc logique que la structure logique correspondante soit aussi séquentielle, ce qui rend les accès et les opérations de recherche très longs.
Ce type de structure est utilisé pour la sauvegarde et l'archivage des informations.

2.2. Structure physique adressée (ou adressable)

La tête de lecture peut se positionner directement à n'importe quel endroit du fichier.
Les fichiers peuvent être fragmentés (découpés en plusieurs parties logiquement chaînées mais physiquement situées à des endroit différents du support).
Cette structure est compatible avec toutes les structures logiques citées.

D. Les organisations et les accès

1. Les organisations

> ***<u>Organisation</u> : Méthode permettant d'attribuer (écrire) un emplacement physique sur le support à tout enregistrement logique***

On peut donc distinguer 4 grands types d'organisations :

1.1. <u>Organisation séquentielle (heap)</u>

Telle une série de fiches dans un bac, sans repères particuliers, une organisation séquentielle est basée sur un fichier séquentiel dont les enregistrements sont répartis sur plusieurs pages de données.
L'impression de désordre que laisse ce type d'organisation explique le nom qui lui est donné (heap veut dire tas).
Les accès sont relativement longs mais si les enregistrements sont triés selon un attribut couramment sollicité (organisation heap triée), cet inconvénient est moins sensible.

Cette organisation a pour avantage de considérer indifféremment chaque attribut de l'enregistrement ; on peut donc accéder aux informations en utilisant comme point d'entrée chaque attribut ou combinaison d'attributs de l'enregistrement, avec des performances équivalentes.

1.2. <u>Organisation à accès direct</u>

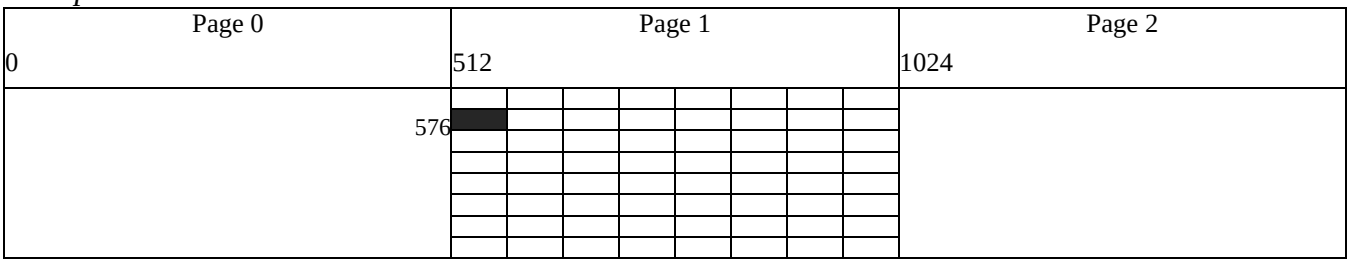
Dans ce type d'organisation, la valeur de la clé d'un enregistrement suffit à retrouver l'emplacement physique de celui-ci.
La zone de données (data file) est découpée en pages.

1.2.1. Adresse physique d'un enregistrement

Plusieurs cas de figure peuvent se présenter :

a) l'adresse de l'enregistrement est l'adresse physique réelle du début de l'enregistrement (le 1er octet) ;
b) l'adresse de l'enregistrement correspond à l'adresse physique du début de la page de données dans laquelle se trouve l'enregistrement recherché ;
c) l'adresse de l'enregistrement est composée de l'adresse physique du début de la page de données et du déplacement qu'il faut effectuer dans la page pour trouver le début de l'enregistrement (**offset**).

Exemple :



L'enregistrement grisé pourrait, selon les trois cas énoncés plus haut, se voir attribuer les adresses suivantes :

a) @ = 576 ;
b) @ = 512 ;
c) @= 512 (*adresse de la page*) + 64 (*offset*).

1.2.2. Mode d'adressage

Ceci étant défini, la valeur de la clé détermine l'adresse physique de l'enregistrement selon deux modes :

a) Adressage direct

La valeur de la clé est la valeur de l'adresse physique de l'enregistrement.

b) Adressage calculé (hash code)

Une fonction mathématique appliquée à la valeur de la clé permet de déterminer la valeur de l'adresse physique.

@ = f (clé)

Exemple : Soit la valeur de clé " DUPONT ", on aura : f (" DUPONT ") ⇨576

1.3. Organisation séquentielle indexée (ISAM)

Il existe plusieurs versions d'organisation séquentielle indexée : IS3 (série 3 d'IBM), ISAM (Indexed Sequential Acces Method), VSAM (Virtual Sequential Acces Method).
Dans le fichier des données, les articles sont rangés séquentiellement à la création selon l'ordre croissant de la clé primaire, et en respectant un certain taux de remplissage par page de données. Dès que cette zone est pleine, les nouveaux articles sont rangés dans une zone de débordement (OVERFLOW) en attendant une restructuration du fichier par programme ou par utilitaire.
La zone de débordement contient les enregistrement ajoutés après la dernière indexation : il faut donc ré-indexer régulièrement le fichier.
En utilisant différents niveaux d'indexation et une indexation partielle des enregistrements, on se retrouve avec des index moins volumineux et tout aussi performants.
Cependant, il faut donc ré-indexer régulièrement le fichier et durant cette opération les informations ne sont pas disponibles.

1.4. Organisation indexée arbre B (B-TREE)

L'organisation B-Tree est aussi appelée arbre balancé (B pour balanced) ou arbre équilibré.
Le principal avantage de cette organisation par rapport à la précédente réside dans la mise à jour dynamique des pages de données et des index, rendant les informations disponibles à tout moment.

Un arbre-B peut être utilisé pour constituer l'index hiérarchisé d'un fichier :

- Il contient des clés triées par ordre croissant et des pointeurs de deux types :
  - ✓ les pointeurs internes qui désignent les branches de l'arbre ;
  - ✓ les pointeurs externes qui désignent des pages de données.
- Il possède au premier niveau une page unique d'index (le tronc), au deuxième niveau plusieurs pages d'index (les branches) et au troisième niveau plusieurs pages de données (les feuilles).
- Chaque entrée d'index donne le numéro de page d'une branche ou d'une feuille.
- Chaque entrée de feuille donne le numéro de page de données et le déplacement dans la page.

Les principes mis en œuvre lors d'un ajout sont les suivants :

Exemple : si la page de données est pleine : elle se coupe en deux ;
on ajoute une entrée dans la page d'index, qui se coupe en deux, si nécessaire ;
si cela n'est pas suffisant, on ajoute un niveau d'index (une branche supplémentaire).

Il n'y a pas de page de débordement, si l'index est unique.

Les mécanismes mis en œuvre font que l'arbre se démultiplie toujours harmonieusement évitant d'ajouter inutilement trop de niveaux d'index.

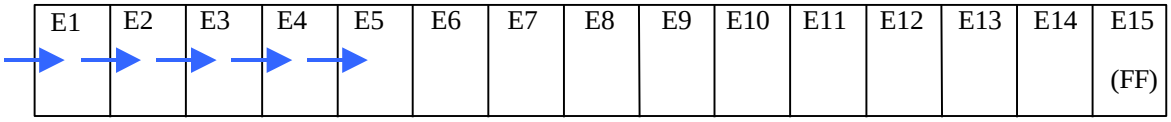
2. *Les accès.*

***Accès** : Façon de trouver (lire) un enregistrement sur le support ; moyen de sélectionner un enregistrement en tenant compte de l'organisation du fichier.*

On distingue 4 grands types d'accès à un enregistrement :

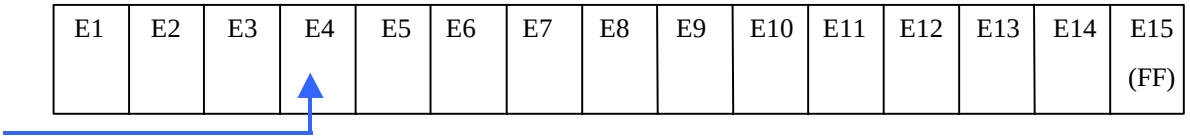
2.1. *Accès séquentiel*

***Accès séquentiel** : accès au premier enregistrement, puis au second …*



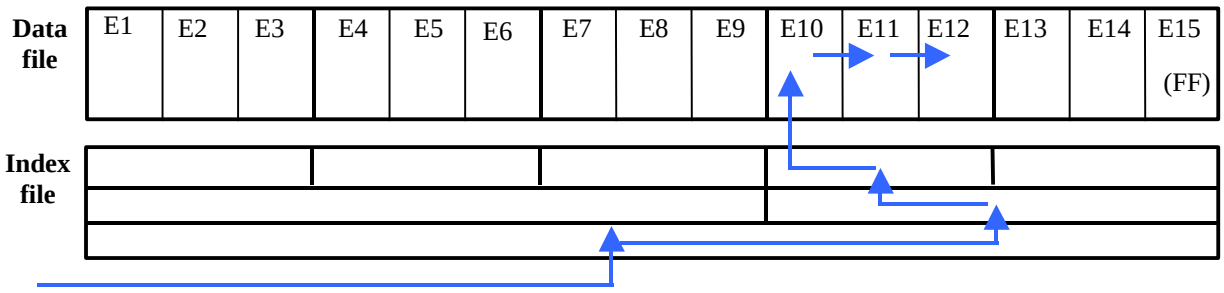
2.2. *Accès direct*

***Accès direct :** l'utilisateur ou le programme fournit une clé à partir de laquelle est calculée l'adresse ou le rang de l'enregistrement.*



2.3. *Accès semi-direct*

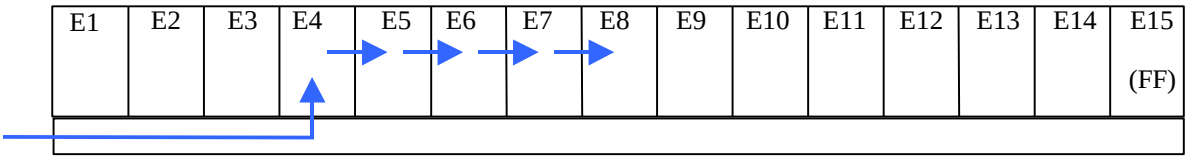
***Accès semi-direct :** l'utilisateur ou le programme fournit la clé et la recherche se fait par l'intermédiaire de table(s) d'index (ou de branches).*



Le fichier des index (index file) est divisé en zone des index de la clé primaire et en zone des index des clés secondaires. Chaque zone permet de localiser (par étapes) l'article recherché.
La table de correspondance (zone index) ne contient que les adresses d'un nombre restreint d'enregistrements en fonction de la rubrique d'index. Ces enregistrements servent de points d'entrée aux pages de données, telles des bornes de recherche.
Pour accéder à un enregistrement, on recherche dans la zone index l'adresse de l'enregistrement le plus proche de celui recherché et on se déplace ensuite séquentiellement dans la page de données.

2.4. Accès dynamique

***Accès dynamique :** on accède à un enregistrement en direct ou en semi-direct et on continue en séquentiel sur les suivants.*



3. Organisations et accès

Accès / Organisation	Séquentiel	Direct	Semi-direct	Dynamique
Séquentiel (heap)				
Direct				
Indexé ISAM				
Indexé BTree				

Accès possible pour un type d'organisation

Sachant comment organiser les différents réceptacles de données que constituent les fichiers, il reste encore à organiser les liens entre ces fichiers, c'est à dire constituer un véritable Système de Gestion de Fichiers qui assure la cohérence et le fonctionnement de l'ensemble.

II. Les Systèmes de Gestion de Fichiers

A. Généralités

Là encore, de fortes similitudes existent entre les systèmes de gestion de fichiers utilisés par les différents systèmes d'exploitation et les systèmes de gestion de fichiers de données, objet de ce chapitre.

1. Fonctionnement d'un SGF

Les fichiers composant le SGF sont des fichiers de données et des fichiers de programmes.
Les fichiers de programmes permettent 2 types de traitements :

> ***traitement interactif :** une action extérieure implique un résultat immédiat de la part du système ;*

ou

> ***traitement par lots :** le traitement comporte une série de commandes qui traitent plusieurs éléments et dont le résultat n'est pas immédiat*

En conséquence de quoi, le traitement interactif est un traitement unitaire et le traitement par lots est un traitement différé.

2. Objectifs d'un SGF

Le Système de gestion de fichiers a pour but, en parfaite collaboration avec le système d'exploitation, d'assurer les tâches suivantes :

- Gérer automatiquement les mémoires secondaires : disques, bandes, mémoire de masse …;
- Gérer les entrées-sorties ;
- Permettre la création et la suppression des fichiers contenant les données ;
- Permettre les accès en lecture et en écriture ;
- Protéger les fichiers contre les défaillances du système (pannes matériel ou logicielles) ;
- Permettre le partage des fichiers entre plusieurs utilisateurs ;
- Protéger les fichiers contre les accès non autorisés ;
- Permettre l'utilisation de langage de programmation pour manipuler et traiter les informations.

B. Les inconvénients des SGF

Au-delà de l'organisation interne de chaque fichier et des mécanismes permettant de manipuler les enregistrements de ces fichiers, la difficulté majeure demeure dans l'existence de différents fichiers au sein du système d'information d'une entreprise ou d'un organisme.

1. Exemple : Gestion des stagiaires de l'ESAT

Si l'on prend le cas du système d'information de l'ESAT, et plus particulièrement l'activité de gestion des stagiaires : les stagiaires sont pris en compte par différents services de l'école, qui doivent, chacun manipuler des informations relatives à leur mission. En conséquence de quoi, chaque service dispose d'un fichier regroupant les informations qui les concernent, et sous une forme qui leur convient :

Service Général :

Grade	Nom	N° Véhicule

BureauPersonnel :

Grade	Nom	Prénom	N° livret de solde

Chancellerie :

Grade	Abréviation grade	Libellé grade	Nom	Prénom

Division d'instruction:

Nom	Prénom	UV	Matière	Note

2. Inconvénients directs

2.1. Redondance des informations

La même information est dupliquée dans plusieurs fichiers où elle peut, en plus, être structurée de manières différentes.
Exemple : Les rubriques grade, nom et prénom.

Cette redondance a pour effet direct une augmentation du volume de l'ensemble des fichiers.

2.2. Inconsistance des données

Les copies d'une même donnée ne concordent pas obligatoirement entre elles.

Exemple : Cas des codifications de grades, orthographes différentes pour le même nom …

2.3. Difficulté à rapprocher les informations

La tâche la plus couramment effectuée, en terme de traitement des données, consiste à associer des informations à d'autres informations, éparpillées un peu partout dans le système d'information de l'entreprise.
Exemple : Retrouver le numéro de livret de solde du stagiaire dont le véhicule porte telle immatriculation, retrouver les commandes d'un client, retrouver les fournisseurs d'un produit …
Ce sont ces associations entre informations qui sont traduites dans un MCD par la mise en place de relations.
Or les données sont de structures différentes et ne sont pas reliées physiquement entre elles, ce qui rend très difficile l'exécution de ces rapprochements : le développeur doit se charger d'effectuer ces rapprochements par comparaison des rubriques des fichiers concernés, en tenant compte de leur organisation, de leurs méthodes d'accès et en espérant qu'il n'y ait pas d'inconsistance !!

3. Inconvénients indirects

Les inconvénients suivants ne sont que la conséquence de ceux précédemment énoncés.

3.1. Difficultés d'accès

Les données étant dupliquées et éparpillées, il y a plusieurs moyens d'y accéder sans savoir à l'avance quel est le mieux adapté au traitement ou à la recherche souhaitée (par quel bout doit-on s'y prendre ?). On a donc besoin d'un système global pouvant s'adapter aux demandes spécifiques et immédiates des utilisateurs.
De plus, il est difficile d'écrire des programmes qui permettent d'accéder aux informations de manière globale ; il devient nécessaire d'établir des relations entre les fichiers.

3.2. Multiplicité des mises à jour

Les données étant dupliquées, la modification d'une information entraîne autant de mises à jour qu'il y d'endroits où cette information est stockée.
Exemple : Un stagiaire féminin du nom de DUPONT se marie et décide de porter son nom d'épouse, DURAND. Il devient obligatoire de modifier la rubrique Nom dans chaque fichier.

Cette multiplicité des mises à jour est aggravée par les difficultés d'accès et par l'inconsistance des données ainsi que par le facteur temps qui rend le système incohérent tant que l'ensemble des mises à jour n'est pas fini.

3.3. Problèmes d'intégrité

Pour garantir un minimum de cohérence au système, on peut mettre en place un certain nombre de règles concernant les données : les contraintes d'intégrité (voir III.B.4 Intégrité et cohérence).
Parmi ces contraintes d'intégrité, on peut citer :

- l'unicité de valeur d'un attribut ;
- le contrôle des valeurs possibles d'un attribut ;
  Exemple : solde >0 ; année naissance < année courante …
- la référence ou appartenance à un ensemble variable de valeurs …

Dans un SGF, la mise en place de ces contraintes est à la charge des programmeurs, qui doivent écrire les programmes correspondants, chose rendue complexe par le contexte d'instabilité et d'incohérence qu'impliquent les inconvénients déjà cités.

De plus, toutes ces difficultés rencontrées augmentent parfois considérablement le temps nécessaire à la mise à jour d'une information sur l'ensemble du système. Celui-ci se retrouve donc régulièrement dans un état incohérent qui peut impliquer des erreurs.
Exemple : En reprenant le cas de figure précédent, si un service ne prend pas rapidement connaissance de la modification de patronyme du stagiaire DUPONT ; ce service ne connaîtra que le stagiaire DUPONT alors que les autres services ne connaîtront que DURAND.

3.4. Problèmes de sécurité

De la même manière, il devient impossible de mettre en place les verrouillages nécessaires pour empêcher une personne d'accéder aux informations dont elle n'a pas le "besoin d'en connaître".

3.5. Indépendance données / traitements

Les langages de programmation utilisés pour manipuler les SGF comportent une très grosse partie dédiée à la déclaration et à la définition des informations traitées ainsi qu'à l'accès et à la manipulation de ces données. Cette lourdeur est démultipliée en cas de maintenance corrective ou évolutive de l'application.
De plus, il est dans l'intérêt d'un propriétaire de données de pouvoir faire évoluer indépendamment les traitements et éventuellement changer le langage de programmation sans remettre en cause toute l'organisation de ces données.

Organiser la gestion de données à partir d'un système de gestion de fichiers est une solution qui n'est pas satisfaisante : lourde à gérer, trop dépendante de l'organisation physique …
Même si l'emploi de méthodes d'analyse permet, dans une certaine mesure, de limiter la redondance et l'inconsistance des données, il reste clair que seules de nouvelles solutions techniques peuvent améliorer la possibilité de rapprocher ou de lier entre elles des informations de même sens.

C'est pourquoi d'autres systèmes de gestion de données se sont mis en place : les Systèmes de Gestion de Bases de Données (SGBD) ; leur objectif principal étant d'éliminer les inconvénients directs des SGF en espérant, par là même, éliminer les inconvénients indirects.

Chaque génération de SGBD constitue une avancée supplémentaire liée à des innovations technologiques ; chaque génération apporte des solutions nouvelles pour atteindre les objectifs énoncés et pour pallier les contraintes de la génération précédente.

III. Notions de SGBD

A. Définitions

1. Base de données

Définition normalisée (BOC/PP du 16/02/81 N7) : Une base de données est un ensemble de données organisé en vue de son utilisation par des programmes correspondant à des applications distinctes et de manière à faciliter l'évolution indépendante des données et des programmes.

Base de données : Ensemble structuré de données, enregistrées sur des supports, accessibles par l'ordinateur, représentant les informations du monde réel et pouvant être interrogées et mises à jour par une communauté d'utilisateurs.

D'autres définitions reprennent les même éléments tout en insistant sur certaines aspects :

- Une base de données est une collection de fichiers (entités) reliés entre eux par des liens logiques et/ou physiques et organisés de manière à répondre efficacement à une grande variété de questions ;
- Une BD peut apparaître comme une collection d'informations modélisant une entreprise du monde réel, servant de support à une application informatique et dont les données peuvent être interrogées au moins par leur contenu ;
- Une BD est destinée à la gestion, au stockage, à l'actualisation et à la consultation d'entités de différentes natures (et de leurs données), sachant que ces entités ont un lien les unes avec les autres.

2. Le Système de Gestion de Bases de Données

On ne gère plus un ensemble de fichiers mais un ensemble de données structurées. En ce sens, le SGBD est donc :

Le système de gestion d'un ensemble cohérent de données non redondantes.

De plus, le SGBD doit répondre aux besoins de toute l'entreprise et non plus d'une application particulière, et ce, dans la limite des droits de chacun. On doit donc aussi considérer le SGBD comme :

Un ensemble de logiciels de gestion, de contrôle d'accès aux données et aux programmes les manipulant.

B. Les objectifs d'un SGBD

1. Centraliser l'information

En visant cet objectif, on cherche naturellement à supprimer la redondance, à assurer l'unicité des saisies et mises à jour et à centraliser les contrôles.

C'est la modélisation conceptuelle qui va permettre, dans un premier temps, de répondre à ce souci.

2. Assurer l'indépendance données - traitements

Il s'agit ici de pouvoir faire évoluer indépendamment données et traitements.

Cette indépendance sera assurée grâce à une approche en plusieurs niveaux de la base de données.

3. Permettre les liaisons entre ensembles de données

On doit pouvoir ainsi établir des liaisons entre ensembles de données qui n'ont que peu de points communs (articles, fournisseurs, clients, comptabilité …).
Dans un premier temps, ces rapprochements seront matérialisés dans la base de données par des liens physiques visibles et manipulables par certains utilisateurs de la base de données. On pourrait alors qualifier cela de "ficelles" existant entre les données. Les versions suivantes de SGBD vont tendre à supprimer ces "ficelles" ; les liaisons entre les données seront assurées par des routines, invisibles mais utilisables à tout moment.

4. Intégrité et cohérence

4.1. Intérêt des contraintes

L'information étant stockée de manière unique, il faut d'autant plus s'assurer de son intégrité, de sa fiabilité et de sa cohérence.
Pour cela, il faut pouvoir définir des contraintes d'intégrité ou des contraintes de cohérence entre données, contraintes qui doivent être prises en compte aussi bien pour la définition que pour le traitement des données et sans faire appel à la programmation (ou le moins possible, pour ne pas invalider les efforts fournis en terme d'indépendance données / traitements).

4.2. Définition

> ***Contrainte d'intégrité* : Règle spécifiant les valeurs permises pour certaines données, éventuellement en fonction d'autres données, et permettant d'assurer une certaine cohérence de la base de données.**

*Exemple : Certaines cardinalités impliquent une contrainte d'intégrité très forte. C'est la cas notamment des cardinalités **1,1**.*

4.3. Typologie

Ces contraintes peuvent prendre les formes suivantes :

- appartenance à une liste de valeurs ou à un intervalle (définition en extension) ;

  Exemple : Les couleurs recensées sont : Bleu, Jaune, Rouge; Les dates de naissance prises en compte sont comprises entre le 01/01/1938 et le 01/01/1972.

- format (nature, longueur ; définition en intention) ;

  Exemple : Les couleurs recensées sont les couleurs primaires, codifiées sur 5 caractères alphabétiques ; Les dates de naissance prises en compte sont codifiées selon le format JJ/MM/SSAA.

- règle de vraisemblance ;

  Exemple : Les dates de naissance prises en compte obligent que l'individu soit majeur mais n'ait pas atteint l'âge de la retraite.

- règle de cohérence avec d'autres informations…

  Exemple : Saisie d'une note pour un élève qui est recensé dans la base de données.

4.4. Modes d'emploi

Il existe plusieurs manières de mettre en place les contraintes au sein d'une base de données :

♦ L'intégrité et la cohérence des informations stockées dans la base de données peuvent être assurées par la mise en place de programmes chargés de vérifier les différentes contraintes lors de création ou de mises à jour d'information. Cette prise en charge par l'applicatif va, bien sûr, à l'encontre de l'indépendance Données-Traitements, alourdit les différents applicatifs de façon importante, avec toutes les conséquences que cela peut avoir sur la maintenance et l'évolution des ces applicatifs.

♦ Les SGBD modernes proposent, eux, de prendre en charge les contraintes, en intégrant ces routines et en les associant directement aux données correspondantes, grâce à **certaines clauses du langage de description de données** employées ou sous la forme de **triggers**.

> ***Triggers : Traitements stockés au niveau du SGBD et déclenchés automatiquement lorsque survient un événement tel qu'une création, une modification ou une suppression.***

5. Partage des données

Les applications doivent pouvoir partager les informations de manière transparente.

Les différentes actions de mise à jour des données doivent pouvoir être effectuées concurremment mais en respectant certaines règles de préséance entre applications et/ou utilisateurs :

Exemple : Si un opérateur sollicite une opération de lecture d'une donnée, cela autorise les autres opérateurs à procéder eux-mêmes à la lecture de cette même donnée, mais interdit toute opération de mise à jour (création, modification ou suppression) de la donnée.
De la même manière toute opération de mise à jour d'une donnée interdit toute autre opération (mise à jour et lecture).

Ces autorisations et interdictions sont gérées par un mécanisme de verrouillage. Cependant, il peut arriver que ces règles ne suffisent pas à réguler les opérations de consultation et de mises à jour des informations, ce qui provoque une situation dite d' interblocage :

> ***Interblocage :*** ***blocage mutuel des deux opérations du fait des mécanismes de verrouillage mis en œuvre.***

6. Sécurité

Il est indispensable d'envisager des procédures garantissant des récupérations contre tout type d'incident (matériel, logiciel), qu'il s'agisse de destructions logiques (anomalies de mise à jour) ou physiques, que ces destructions soient partielles ou totales.
Avant de présenter les mécanismes qui assurent cette sécurité, il est nécessaire de définir les notions de transaction et de requête.

> ***Requête : Unité élémentaire de traitement permettant d'agir sur un SGBD.***

Exemple : - Lecture des toutes les informations relatives aux clients ;
- Création d'un nouveau client ;
- Création d'un utilisateur du SGBD …

> ***Transaction : Ensemble de 1 à n requêtes nécessaires à la réalisation d'une opération particulière.***

Exemple : - Retrait d'argent d'un compte avec vérification préalable du solde ;
- Enregistrement de la livraison d'un produit avec mise à jour de la quantité en stock …

6.1. Atomicité des transactions

Pour que ces opérations soient réalisées de manière fiable en obtenant des résultats cohérents, il est indispensable que l'intégralité des requêtes constituant une transaction soit exécutée et que les résultats définitifs soient enregistrés dans la base de données.

En fait, un SGBD devra veiller à ce qu'une base de données reste dans un état cohérent. Pour cela les transactions ne doivent pas être exécutées partiellement : elles doivent être réalisées, soit entièrement, soit pas du tout. D'où le qualificatif **d'atomique** (qui constitue un tout indissociable).

Exemple : Si on considère une opération de transfert d'un montant M d'un compte bancaire C1 sur un compte bancaire C2, on peut concevoir que cette transaction nécessite les requêtes suivantes :

Requêtes	Actions	Résultats
R1	Lecture du montant du solde	Solde C1 = Solde C1
R2	Débit sur C1	Solde C1 = Solde C1 - M
R3	Crédit sur C2	Solde C2 = Solde C2 + M

Si un incident survient entre les requêtes R2 et R3, la base de données est dans un état logiquement incohérent : le débit est effectué sur C1, le solde de C2 est erroné, le montant M a disparu !!

Une grande partie du fonctionnement d'un SGBD est guidée par le respect de cette contrainte d'atomicité des transactions.

6.2. Validation et annulation des résultats d'une transaction

6.2.1. La commande COMMIT

Lorsqu'une transaction s'est déroulée correctement et dans son intégralité, plus rien n'empêche que les résultats soient enregistrés définitivement. Cette validation des résultats est assurée par la commande :

> ***Commit : Commande exécutée par le système, qui rend effectifs les résultats d'une transaction.***

Cette commande est aussi mise à la disposition des utilisateurs qui ont la possibilité d'en demander l'exécution après chacune de leurs requêtes. De plus, l'exécution de cette commande libère les verrous posés par chacune des requêtes de la transaction.

6.2.2. La commande ROLLBACK

Si une transaction ne s'exécute pas complètement, le système dispose de la commande **ROLLBACK** lui permettant d'annuler les effets partiels de la transaction :

> ***Rollback : commande exécutée par le système, qui annule les résultats d'une transaction.***

Comme pour la commande COMMIT, l'utilisateur a la possibilité d'employer cette commande, pour éviter, par exemple, qu'une opération malheureuse ne vienne endommager la base de données (suppression intempestive, annulation de toutes les requêtes en cas de message d'erreur …). La commande ROLLBACK libère également les verrous posés par les différentes requêtes.

Exemple : Opération de crédit d'un montant de 100,00 FF sur un compte C1 :

Requêtes	Actions	Résultats
R0	**Commit**	Solde C1 = 0
R1	Lecture du montant du solde	Solde C1 = 0
R2	Crédit sur C1	Solde C1 = 100
R3	Lecture de C1	Solde C1 = 100
R4	**Rollback**	Solde C1 = 0 , l'effet de la requête R2 est annulée !

6.3. Les mécanismes de sécurité

6.3.1. Les points de reprise

Lorsque le SGBD redémarre après un incident, il doit impérativement le faire à partir d'une situation de référence dans laquelle les données des bases de données sont dans un état stable. Pour garantir cela, le SGBD effectue à intervalles réguliers des points de reprise.

> ***Point de reprise : Etat stable créé par le SGBD à intervalles réguliers. Le SGBD effectue un bilan des transactions en cours et valide les transactions terminées (les résultats de ces transactions sont écrits sur disque)***

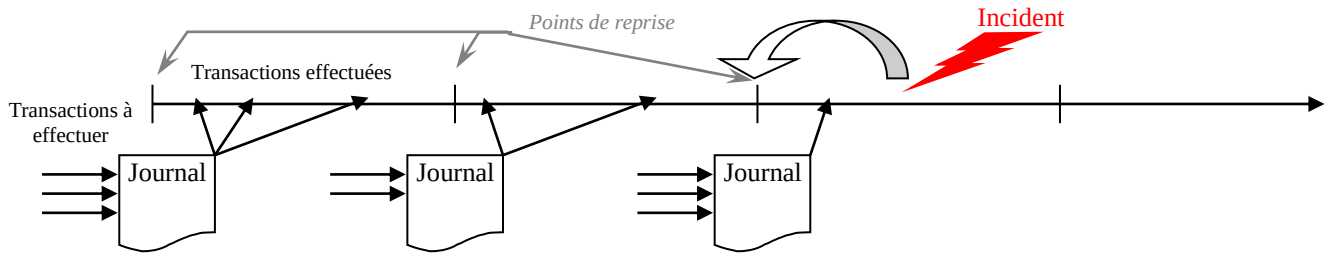
6.3.2. Le journal des transactions

A l'issue du point de reprise, toutes les transactions sont enregistrées dans un journal :

> ***Journal des transactions : Fichier mémorisant l'état des bases de données à l'issue du dernier point de reprise ainsi que toutes les transactions effectuées par le SGBD depuis ce dernier point de reprise.***

6.3.3. La procédure de reprise

Lors de la reprise après incident, le système redémarre depuis le point de reprise précédent en appliquant le journal des transactions, pour se retrouver dans la même situation qu'avant l'incident : il s'agit d'une **procédure de reprise.**



6.4. Duplication des données

Pour se protéger des destructions physiques massives d'informations, seules les duplications (sauvegardes) par des dispositifs réguliers (hebdomadaires, journaliers…) ou permanents (mirroring, réplication…) constituent une garantie suffisante.

7. Confidentialité

Un SGBD doit offrir une protection des données afin d'éviter les accès illicites.

On peut notamment assurer la confidentialité en mettant en œuvre des procédures :

- **d'identification** ;

  Exemple : nom d'utilisateur ou login

- **d'authentification** ;

  Exemple : mot de passe

- **d'autorisation d'accès**.

  Exemple : définition des possibilités de consultation de création, de modification, de suppression d'une ou plusieurs entités au profit des utilisateurs.

N.B. : Le souci d'identification et d'authentification pour accéder à un SGBD se superpose systématiquement au même besoin pour le système d'exploitation.
Pour éviter qu'un utilisateur ne soit obligé de s'identifier et de s'authentifier deux fois (une fois pour accéder à sa machine et une fois pour accéder au SGBD), il peut être possible de reprendre les éléments d'identification et d'authentification du SE (même nom d'utilisateur, même mot de passe). Ce dispositif s'appelle le "**mapping**".

IV. Les SGBD de 1ère génération

Vers 1965 des sociétés comme Honeywell (IDS 1) et IBM (IMS 1) ont développé des produits permettant de constituer des chaînes d'articles entre les fichiers et de parcourir ces chaînes : la multi-indexation. On retrouve derrière cette innovation la volonté de faciliter le rapprochement d'informations.
A la fin des années 60, les premiers SGBD apparurent accompagnés des premiers langages navigationnels

A. Principes

L'innovation principale réside dans la séparation de la description et de la manipulation des données.
L'apparition de langages permettant de manipuler les données va résoudre en partie le problème de la redondance et de l'inconsistance des données. Ces langages utilisent les liens logiques et physiques mis en place lors de la construction de la base de données.

B. Les modèles hiérarchiques

1. Innovation

Ce type de SGBD utilise les possibilités suivantes :

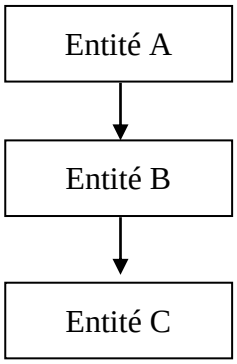
1.1. La représentation graphique des données

La dépendance entre les données est formalisée selon des règles simples :

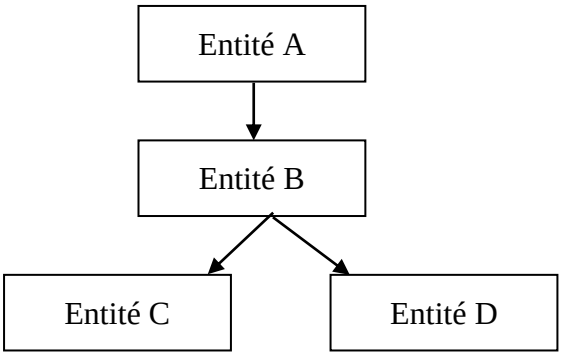
> ***Modèle linéaire : chaque entité a une seule entité 'mère" et, au plus, une seule entité "fille".***
>
> ***Modèle arborescent : Chaque entité a une seule entité "mère" mais peut avoir plusieurs entités "fille".***

Bien sûr ces règles ont un impact direct sur la représentation graphique des modèles de données :

Modèle hiérarchique linéaire



Modèle hiérarchique arborescent

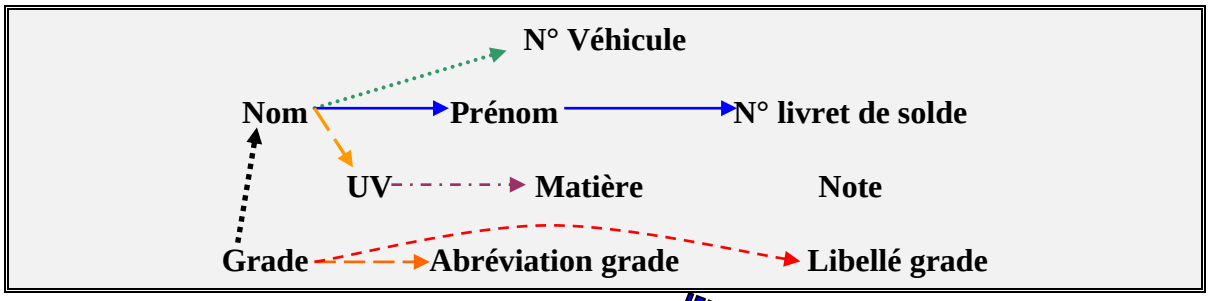


Dans les deux cas, l'organisation reste sous la forme d'arborescence, mais des liaisons entre les éléments d'un même niveau ne sont pas possibles, d'où le terme de hiérarchique (*Il n'existe pas de lien logique direct entre les entités C et D*).

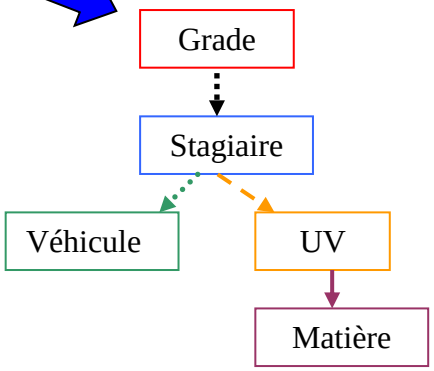
Exemple : En reprenant l'exemple "école", on peut d'abord éliminer l'ensemble des données redondantes et/ou inconsistantes sur l'ensemble des fichiers :

Service Général (~~Grad, Nom~~, **N° Véhicule**)
Bureau Pers (~~Grade,~~ **Nom, Prénom, N° livret de solde**)
DMSI (~~Nom, Prénom~~, **UV, Matière, Note**)
Chancellerie (**Grade, Abréviation grade, Libellé grade,** ~~Nom, Prénom~~)

On peut ensuite mettre en évidence un certain nombre de dépendances entre les données restantes :



Et modéliser la base de données de la manière suivante :



1.2. L'utilisation de pointeurs

La dépendance logique entre les données est matérialisée par la mise en place de liens physiques : les **pointeurs**.
Ces pointeurs sont différents des pointeurs utilisés en programmation :

- ils sont définis à la création de la base de données ;
- la valeur de ces pointeurs est stockée en mémoire de masse, au même titre que les données vers lesquelles ils pointent.

Ainsi chaque occurrence d'entité dispose d'un pointeur vers une autre occurrence, que celle-ci appartienne ou non à la même entité.
Les liens logiques et les liens physiques sont donc presque confondus.

1.3. Les chemins d'accès aux données sont prédéterminés

La mise en place des pointeurs détermine rigoureusement les cheminements possibles pour passer d'une information à une autre.

2. Concepts associés

2.1. Vocabulaire

2.1.1. Record

Record (ou Segment) : Occurrence d'une entité.

Exemple : Le record composé des données "MARTIN", "Jean-paul", "6411280" est une occurrence de l'entité Stagiaire.

2.1.2. Set

Set (ou Lien) : Lien existant entre des segments issus d'entités différentes.

Exemple : Le lien existant entre les records "SCH" et "MARTIN, Jean-paul, 6411280"

2.1.3. Père/fils ou Mère/fille

Dans le cas d'un SGBD hiérarchique arborescent, on obtient les combinaisons suivantes :

- A une occurrence d'un segment "père" on peut associer plusieurs occurrences du segment "fils" ;
- A une occurrence du segment "fils" ne correspond qu'une occurrence du segment "père".

On dit alors que le segment "père" est propriétaire (ou owner) du segment "fils", alors que le segment "fils" est membre (ou member) du segment père.

Sur le lien, le sens de cette relation est symbolisé par :

- la flèche, qui part du "père", pour aller au "fils" ;
- par le nom du lien : composé des 3 premiers caractères du segment père puis des 3 premiers caractères du segment fils .

Exemple : Le set compris entre Grade et Stagiaire s'appelle GRASTA.

2.1.4. Pointeur

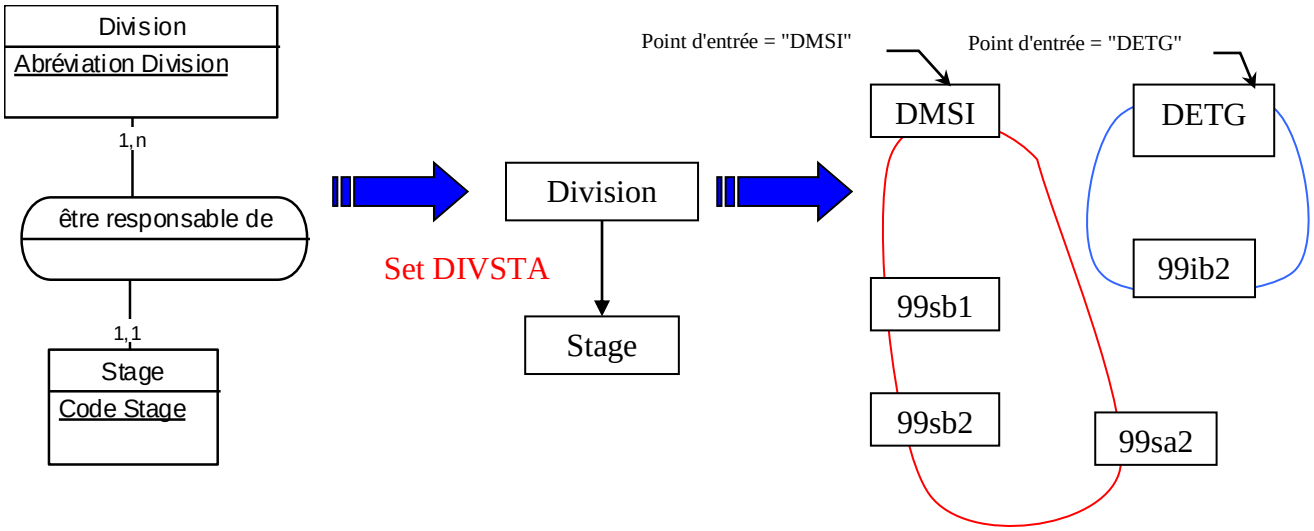
A chaque segment est donc associé un pointeur qui matérialise le lien avec le segment "père" ou le segment "fils".

2.1.5. Point d'entrée

Point d'entrée : Possibilité offerte par le SGBD d'accéder directement à une occurrence d'une entité.

Dans les SGBD linéaires et hiérarchiques, il n'existe qu'un seul point d'entrée dans toute la base de données : le sommet de l'arbre.

2.2. *Application au cas Division-Stage*



Modèle Conceptuel des Données Modèle Logique des Données Représentation de quelques segments et de leurs liens

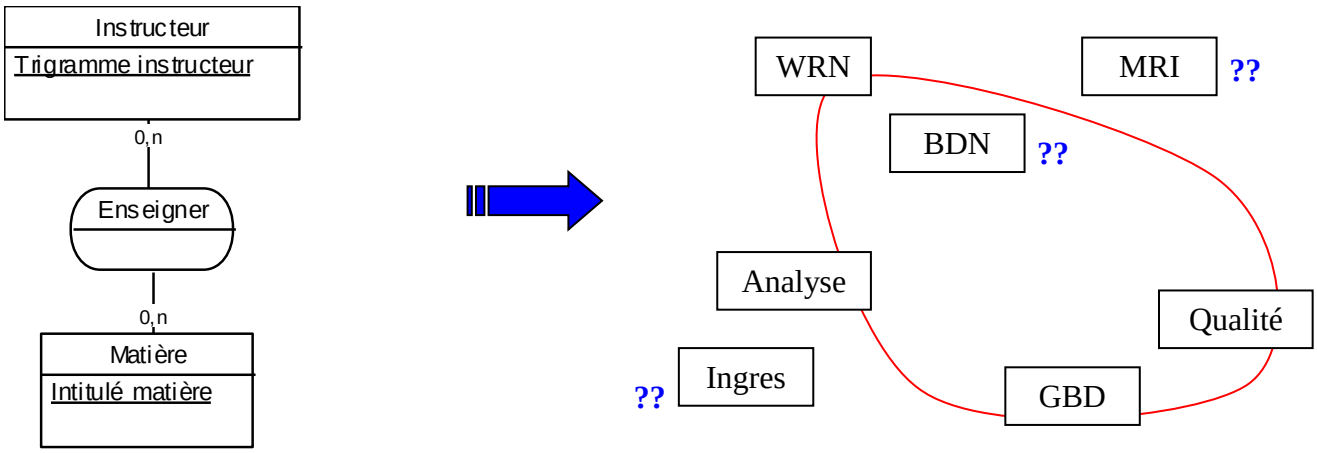
3. *Avantages*

Grâce aux SGBD de 1ère hiérarchiques :

- La dépendance et la hiérarchie entre les données est conservée ;
- On peut parler de cheminement logique au sein des données ;
- On utilise un langage de manipulation de données qui assure une meilleure indépendance entre les données et les traitements. Ce langage est "navigationnel".

4. *Inconvénients*

- On ne bénéficie que d'**un seul point d'entrée** dans la base de données.
- La structure induite est contraignante car **elle influence fortement le choix d'accès aux données et l'organisation physique des données** et cette rigidité n'est pas compatible avec l'évolution constante d'un système d'information.
- Ces SGBD étant conçus pour exploiter des liens pré-déterminés, l'analyse préalable doit donc être suffisamment exhaustive pour envisager tous les rapprochements d'informations souhaitables et les traduire par des liens physiques.
- Les **cardinalités minimales égales à 0 ne peuvent être prises en compte** par ces SGBD (une occurrence en désigne toujours une autre).
- De la même manière, on ne peut traiter le cas des relations portant, de part et d'autre, des cardinalités maximales égales à n et notamment les relations de type n-aire.



C. Le modèle réseau (dit norme CODASYL)

Le SGBD réseau se conforme aux normes fixées par le groupe CODASYL (Conférence On Data System Languages) en 1971. Les SGBD réseau les plus représentés sur le marché sont les systèmes IDS II (Information Data Store), conçu par BACHMAN et WILLIAMS, ou IMS 2 d'IBM.

1. Innovation

Ce type de SGBD propose de trouver une solution pour :

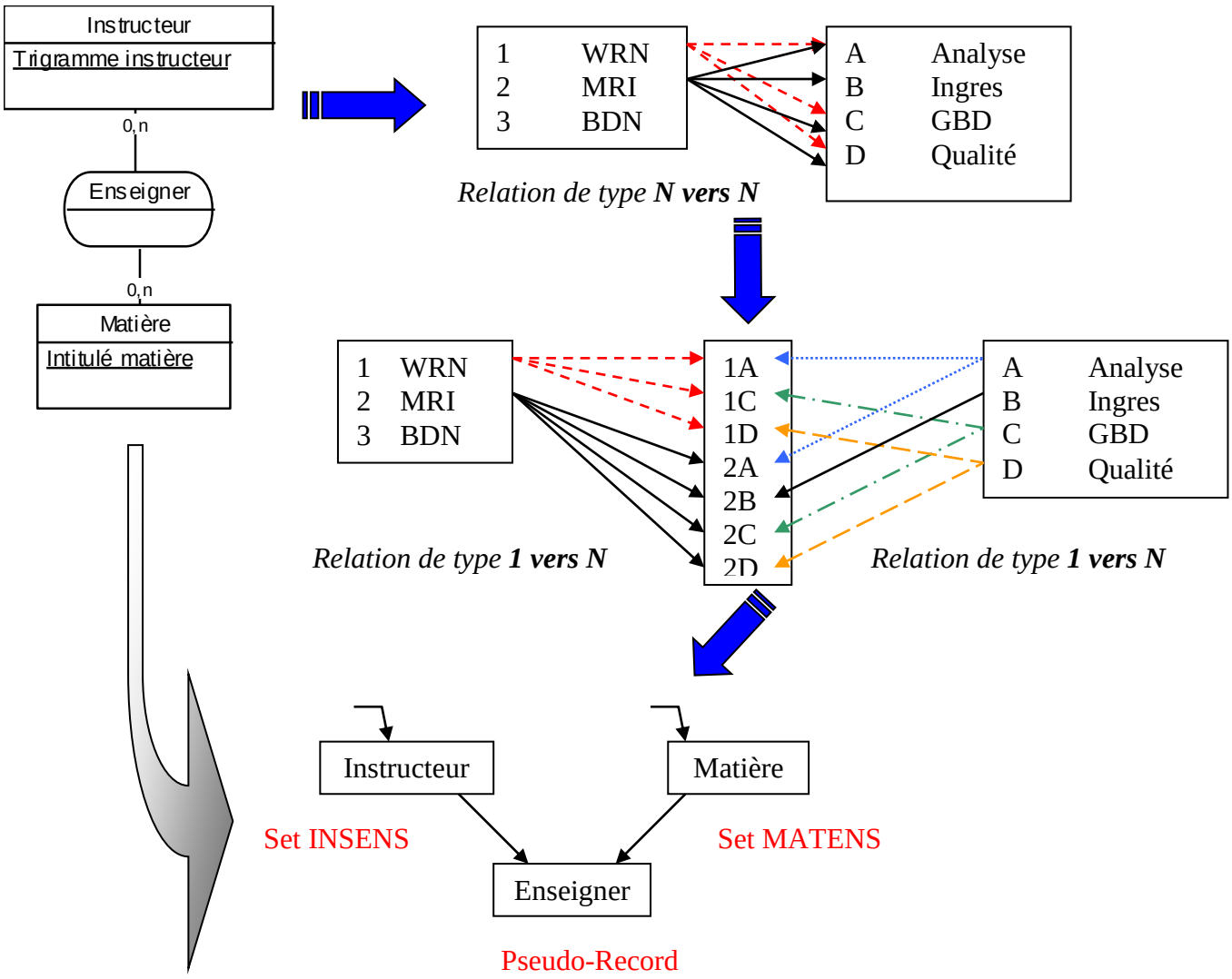
- gérer les relations porteuses de cardinalités maximales à n ;
- prendre en compte la cardinalité mini à 0 (sous certaines conditions);
- gérer les relations de type n-aire ;
- obtenir plusieurs points d'accès, autres que le sommet de l'arbre.

Pour cela, il modifie l'une des règles de dépendance entre les entités :

Une entité "fille" peut avoir plusieurs entités "mères".

2. Présentation

2.1. Démonstration



Représentation de Bachman dite "shorthand"

Le problème des relations à cardinalités maximales fixées à n est donc réglé en permettant la mise en place de records "intermédiaires", qui servent de "relais" entre chacun des records dont il est le membre.

2.2. Les jeux de pointeurs

De plus, les SGBD CODASYL permettent la mise en place des pointeurs, non pas en fonction des segments mais en fonction des sets (liens) portés par les segments :

- A un segment propriétaire d'un set, on attribue un jeu de pointeurs dits **F**(irst) et **L**(ast) ;
- A un segment membre d'un set, on attribue un jeu de pointeurs dits **N**(ext), **P**(rior) et **O**(wner).

Le pointeur First pointe sur la première occurrence du set ;

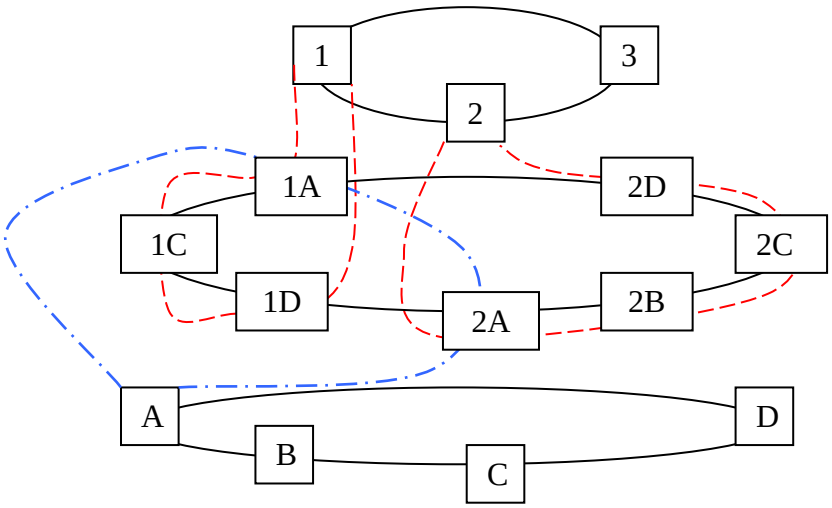
Le pointeur Last pointe sur la dernière occurrence du set ;

Le pointeur Next pointe sur l'occurrence suivante dans le set ;

Le pointeur Prior pointe sur l'occurrence précédente dans le set ;

Le pointeur Owner pointe sur l'occurrence propriétaire du set.

Exemple : Application au cas précédent Instructeur/Enseigner/Matière :



1	
F : @1A	L : @1D

1A		
N : @1C	O : @1	P : @1

Exemple de valeurs de pointeurs pour le set INSENS appliqué à l'instructeur 1

Représentation de BACHMANN développée dit aussi représentation "longhand"

2.3. Les pseudo-records

Emanation directe des différentes innovations proposées par le modèle réseau, un nouveau type de record apparaît : le pseudo-record.

Pseudo-record : Record intermédiaire qui ne contient que les différents jeux de pointeurs des records dont il est membre. Il n'est donc pas porteur d'information.

Exemple : Application au cas précédent Instructeur/Enseigner/Matière : Le record Enseigner, résultant de la relation du même nom, porteuse de cardinalités maximales à n, ne contient que les valeurs des différents pointeurs des sets INSENS et MATENS ; c'est donc un pseudo-record.

2.4. Les points d'entrée

Enfin les SGBD de type CODASYL permettent de mettre en place d'autres points d'entrée dans la base de données : les Data-Record-Key. Ces clés d'accès sont positionnées sur les segments régulièrement sollicités pour certains traitements, ce qui permet d'améliorer les temps d'accès aux données et d'alléger les traitements de recherche de ces mêmes données.

3. Avantages

Il est clair que ce type de SGBD apporte une plus grande souplesse et une plus grande rapidité aux différents traitements.

4. Inconvénients

- Les chemins d'accès aux données restent très dépendants de la structure adoptée.
- Le langage navigationnnel qui permet d'accéder et de manipuler les données reste lourd : il faut connaître le chaînage et les jeux de pointeurs mis en place, en plus de la signification des données.
- En dehors de traitements préétablis, la base de données n'est accessible qu'à des spécialistes.
- Le schéma de la base de données, avec les jeux de pointeurs et les clés d'accès aux records, nécessite une compilation à chaque modification de structure et la refonte des différents programmes. L'évolution ou la modification de telles bases de données restent donc délicates à mener.

5. Passage du modèle conceptuel des données au modèle logique "CODASYL"

Au cours de l'analyse préalable à la mise en place d'une base de données, les concepteurs utilisent pour modéliser les données du système d'information les concepts et les règles du MCD. Le modèle doit ensuite être transformé pour préparer son implantation physique sur les structures de stockage : on obtient alors le Modèle Logique de Données, dépendant du type de structure de stockage des données employé. Pour réaliser ce passage du MCD au MLD, il existe un certain nombre de règles édictées en fonction de la nature de la relation conceptuelle existant entre un ou plusieurs objets.

5.1. La transformation des objets du MCD

Dans tous les cas de figure :

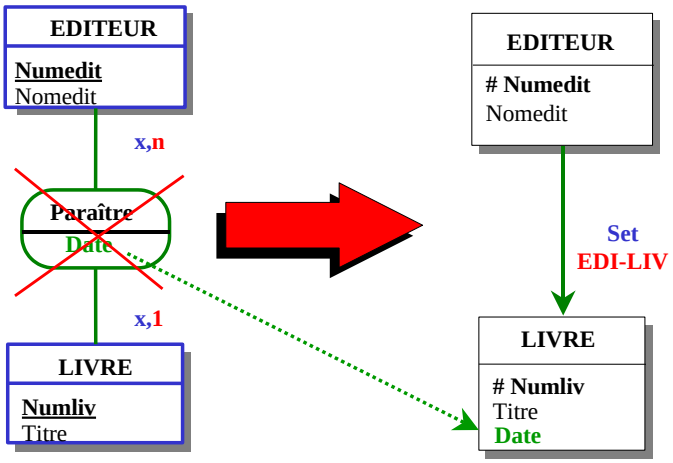
- **Les objets sont transformés en SEGMENTS (ou RECORDS).**
- **Les propriétés des objets sont stockées dans les segments ainsi créés.**

5.2. La transformations des relations conceptuelles

5.2.1. Cas des relations dites "de 1 vers N" :

- **Une relation conceptuelle porteuse de cardinalités maximales égales à 1 d'un coté et à N de l'autre, devient un SET.**
- **Le segment propriétaire (père) de ce set est issu de l'objet qui portait la cardinalité maximale à N.**
- **Le segment membre (fils) de ce set est issu de l'objet qui portait la cardinalité maximale à 1.**
- **Les propriétés portées par la relation conceptuelle migrent vers le segment membre (fils).**

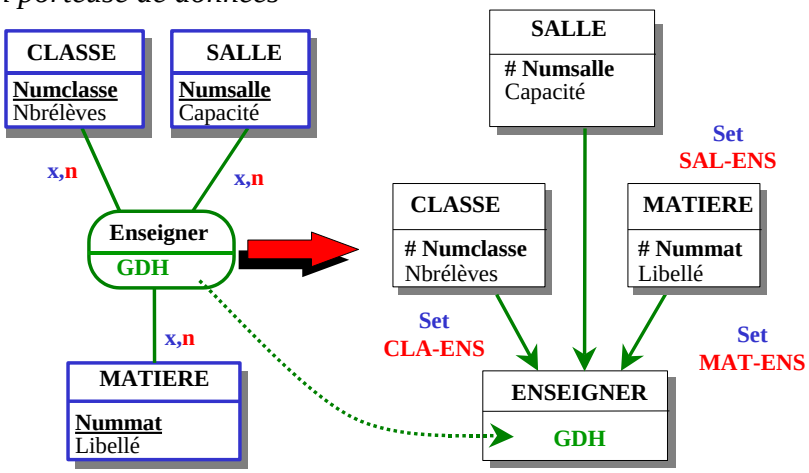
Exemple : Relation porteuse de propriétés



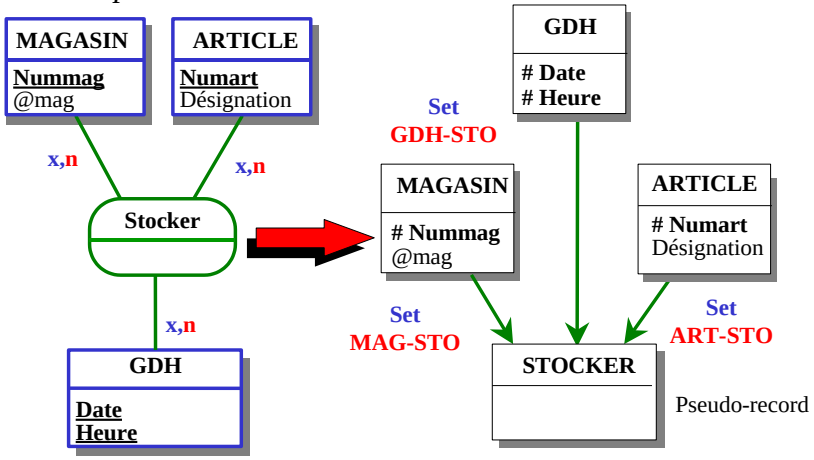
5.2.2. Cas des relations dites "de N vers N" :

- **Une relation conceptuelle porteuse de cardinalités maximales égales à n sur chacun de ses liens, qu'elle soit ou non porteuse de données, se transforme en segment intermédiaire, membre (fils) de chacun des segments à l'origine de la relation.**
- **Si la relation conceptuelle est porteuse de données, celles-ci sont stockées dans le segment intermédiaire ainsi créé.**
- **Si la relation conceptuelle est exempte de toute propriété, le segment intermédiaire est alors qualifié de pseudo-record.**

Exemple : Relation porteuse de données



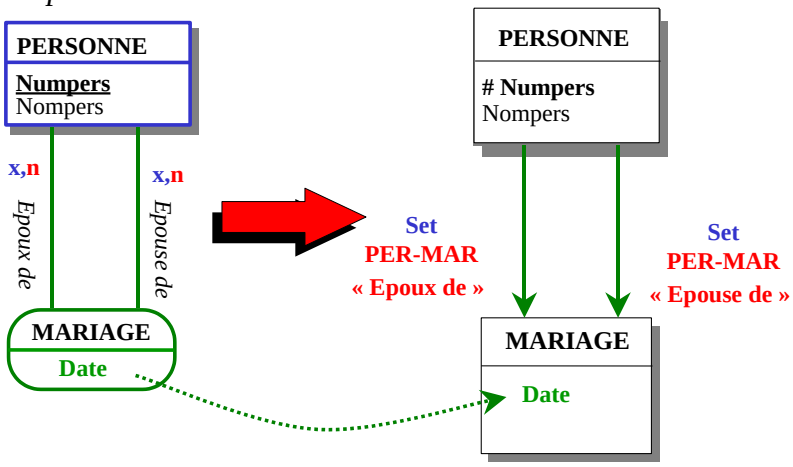
Exemple : Relation non porteuse de données



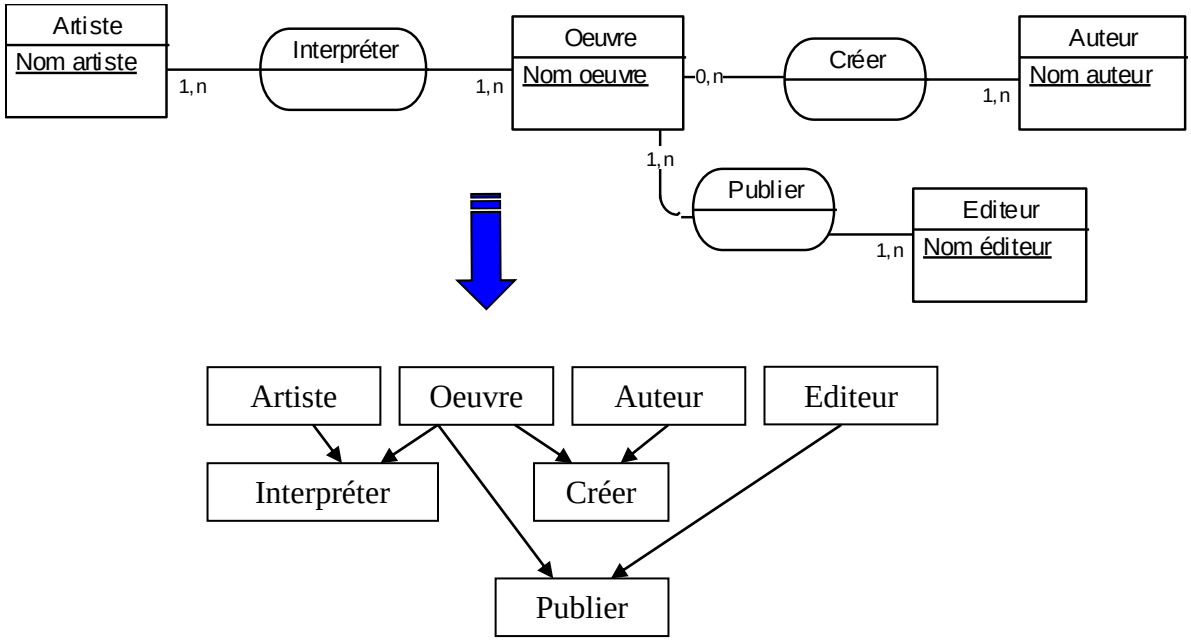
5.2.3. Cas des relations réflexives

- **Une relation conceptuelle réflexive, qu'elle soit ou non porteuse de données, se transforme en segment intermédiaire, membre (fils) du segment à l'origine de la relation, par le biais de deux sets, représentant chacun un sens de lecture de la relation réflexive d'origine.**
- **Si la relation conceptuelle est porteuse de données, celles-ci sont stockées dans le segment intermédiaire ainsi créé.**
- **Si la relation conceptuelle est exempte de toute propriété, le segment intermédiaire est alors qualifié de pseudo-record.**

Exemple : Relation porteuse de données



6. *Application*



V. Les SGBD Relationnels

Ce type de SGBD est commercialisé depuis 1982, mais connaît un véritable boom au début des années 90, certains d'entre eux étant intégrés comme logiciels de bureautique dans les différentes suites logicielles proposées par les principaux éditeurs.

Quelques SGBD relationnels, parmi les plus connus : INGRES, ORACLE, SYBASE, INFORMIX, DB2, ACCESS.

Les SGBD de 2ème génération marquent une évolution de la mécanique de rapprochement d'informations.

S'appuyant sur l'application de la notion ensembliste de relation (différente des relations utilisées dans les MCD), objet des travaux du mathématicien CODD depuis 1964, ce type de SGBD propose :

- une structure relationnelle de stockage des données ;
- la disparition de la représentation en graphe des données (représentation de BACHMAN) ;
- la suppression des "ficelles" physiques liant les données entre elles ;
- la facilitation de l'accès aux données pour les utilisateurs, notamment non-informaticiens.

Relation : Tableau de données à 2 dimensions.

Les relations se manipulent à l'aide de l'algèbre relationnelle, véritable support mathématique.

Le rapprochement entre données est possible grâce à l'existence de **domaines communs** dans l'ensemble des **relations** (tables) de la base de données.

Les recherches et mises à jour sont effectuées à l'aide d'un langage **non procédural et non navigationnel** permettant de spécifier les données que l'on veut obtenir sans dire comment y accéder, ainsi que de traitements ensemblistes. C'est le SGBD qui doit déterminer le meilleur plan possible d'accès aux données.

VI. Les SGBD Objets et les autres formes de SGBD

A. Les SGBD Objets

La troisième génération de SGBD est en développement dans les laboratoires depuis le début des années 1980. Outre l'enrichissement des fonctionnalités des SGBD/R, elle est également fondée sur les BD Orientées Objet, qui répondent aux besoins des nouvelles applications (CAO, Bureautique).
Citons quelques SGBDOO déjà commercialisés : ORION, O2, VERSANT.

B. Les nouvelles formes de SGBD

1. Les bases de données réparties

Jusqu'au début des années 1980, le coût élevé des ordinateurs a favorisé la notion de bases de données centralisée : un seul endroit pour stocker et administrer les données de l'entreprise.
Le développement des réseaux de communication, des mini et micro-ordinateurs, ainsi que des bases de données relationnelles a permis à la notion de base de données répartie de devenir une réalité.

Simplement distante ou répartie, homogène, ou hétérogène, fédérée ou parallèle, chaque structure apporte sa solution à une situation spécifique.

2. Les info-centres

Dans toute entreprise, l'information existe sous différentes formes, sur différents supports. Or, au niveau le plus haut de l'entreprise existe un besoin décisionnel qui nécessite une information synthétique agrégée, basée sur l'exploitation des informations élémentaires.
Un info-centre est un ensemble d'outils logiciels capables d'extraire et d'exploiter des informations élémentaires d'une ou plusieurs bases de données et de les présenter sous une forme synthétique attrayante. Ce n'est pas un SGBD.

L'armée de terre a retenu Business Object (de la société Business Object) alors que la Marine Nationale utilise dans de nombreux endroits Impromptu (de la société Cognos).

VII. Les acteurs d'un SGBD

On peut caractériser les utilisateurs d'un SGBD en 3 grandes catégories :

A. Les administrateurs

1. L'administrateur système (ou root ou administrator)

Il est le responsable de la configuration du système d'exploitation et du matériel qui accueillent le SGBD. Il attribue les ressources système à l'administrateur du SGBD.

2. L'administrateur du SGBD (ou sa : system administrator)

Il est responsable de :
- l'installation ;
- l'attribution des droits aux différents usagers ;
- performances du SGBD ;
- la maintenance du SGBD…

3. L'administrateur de base de données (ou dba : data base administrator)

Responsable d'une ou plusieurs bases de données, il est :
- propriétaire de tous les privilèges au sein de ces bases de données ;
- responsable :
  - des performances des structures de données ;
  - de la sécurité (droits d'accès) ;
  - de l'intégrité ;
  - de la maintenance ;
  - des sauvegardes …

C'est le responsable de la base de données, de sa création et de sa maintenance. Il dispose d'un ensemble d'outils logiciels qui permettent de l'assister dans sa tâche.
Son rôle :

- Décider des techniques d'accès et d'implantation physique ;
- Etablir la liaison SGBD - Utilisateur en définissant les schémas externes ;
- Assurer le contrôle des données et leur intégrité ;
- Mettre en oeuvre la stratégie de reprise en cas d'incident ;
- Mesurer les performances du système (au niveau de la base) et réaliser en conséquence les modifications de structure.

4. L'administrateur des données

Propriétaire des données stockées dans la base, l'administrateur des données n'est pas un informaticien mais un fonctionnel du domaine concerné. Il est souhaitable qu'il ait participé à la conception de la base de données. Il veille tout particulièrement à la cohérence et au codage des tables de référence et dispose, à ce titre de privilèges particuliers.

Exemple : Il est le seul habilité à créer et codifier un nouveau grade, un nouveau service, un nouveau personnel, ... dans la base de données.

B. Les utilisateurs" informaticiens"

1. Les programmeurs d'applications

Les programmeurs et concepteurs d'application conçoivent les traitements en connaissant tout ou partie de la base de données et en utilisant des langages hôtes ou des outils du SGBD.

2. Les utilisateurs occasionnels

Ces utilisateurs se servent plus particulièrement du langage de requête pour effectuer des opérations de consultation et / ou de mises à jour ponctuelles, en dehors de tout environnement applicatif.

C. Les utilisateurs non informaticiens (ou exploitants)

Ils effectuent des opérations élémentaires sur la base de données en utilisant des programmes définis. Les moyens et mécanismes employés leur sont inconnus, le SGBD et la BD leur sont transparents, ils n'ont qu'une vision partielle des données et des traitements afférents.

Pour les "petits" SGBD et BD, les différents rôles d'administrateurs et d'utilisateurs informaticiens sont souvent confondus.

Des Systèmes de Gestion de Fichiers aux puissants info-centres la même problématique demeure malgré l'évolution des techniques et des modèles de représentation.
On dispose désormais de toute une panoplie d'outils pour gérer les données vitales d'une entreprise. L'enjeu est de garder à l'esprit les avantages et inconvénients de ces différentes architectures pour choisir la plus judicieuse, lors d'un choix d'architecture.
Même si l'architecture la plus présente actuellement est celle des SGBD/R, il faut savoir manipuler les concepts associés à chacune des architectures présentées et être capable de passer de l'une à l'autre ou d'un modèle de représentation de l'une vers celui d'une autre.