LABORATOIRE DE ZOOÉCOLOGIE DU SOL INRA/CNRS CEPE

DONNEES ECOLOGIQUES ET SYSTEME RELATIONNEL DE BASES DE DONNEES

PREMIÈRE PARTIE : NOTIONS ÉLÉMENTAIRES POUR LA CONSTITUTION DE LA BASE DE DONNÉES "ECORDRE"

MONTPELLIER, MAI 1985 PATRICIO SOTO
VERSION 1.0

T A B L E

DONNEES ECOLOGIQUES ET SYSTEME RELATIONNEL DES BASES DE DONNEES

LA NOTION DE SYSTEME D'INFORMATION

QU'EST-CE QU'UN SYSTEME ?

Pour éliminer au mieux les ambiguités sur ce terme trop souvent employé, il apparaît préférable de fournir des définitions complémentaires du mot système :

a) Un système est un ensemble d'éléments pouvant chacun revêtir divers états; l'état du système est alors représenté par la liste des états de chacun des éléments qui le composent.

b) Un système est un ensemble de variables pouvant prendre diverses valeurs; l'état du système est représenté par la liste des valeurs de variables.

c) Unité formée de plusieurs parties, souvent diverses, assujetties à un plan commun ou servant un but commun.

Toutes ces définitions essaient de traduire la dualité du concept "système", défini à la fois:

---par ses objectifs, c'est alors un ensemble de tâches à exécuter dans de conditions données.
---par sa composition, c'est alors un ensemble organisé d'éléments fonctionnels.

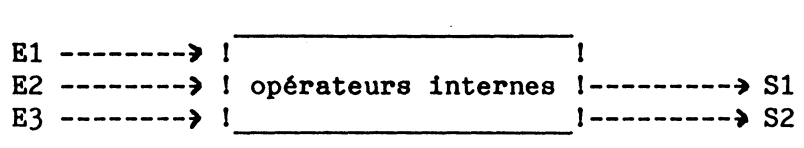
Par exemple:

Un système de chauffage peut être défini par ses éléments (chaudière,radiateurs, pompe, thermostat...) et par ses objectifs (assurer une température de 22° à l'intérieur lorsque la température externe est de -5°).

COMMENT EST COMPOSE UN SYSTEME ?

On distingue dans un système: les entrées, les sorties et les opérateurs.

---LES ENTREES: ce sont les variables dont les valeurs sont imposées au système par l'extérieur.
---LES SORTIES: ce sont les variables qui agissent sur l'extérieur, c'est-à-dire, sur d'autres systèmes. Aussi, certaines de ces sorties ont pour objet de renseigner sur l'état du système ou l'efficacité de son fonctionnement.
---LES OPERATEURS: ils réalisent la transformation des entrées en sorties. Cette transformation représente la fonction même du système.



composition d'un système

Par exemple:

Système "d'enseignement": les entrées sont des étudiants (niveau A), des programmes; les sorties sont des étudiants de niveau B (B normalement supérieur à A). Les opérateurs (enseignants, moyens audiovisuels, etc.) assurent la transformation. Dans ce cas, on peut parler d'opérateurs sociologiques car il est difficile de formaliser la transformation niveau A en niveau B.

QUELLE EST LA SIGNIFICATION DU TERME INFORMATION DANS UN SYSTEME?

Dans l'expression "système d'information", le terme "information" peut être assimilé à "renseignement" ou "connaissance"; il évoque un concept, un contenu, un signifié.

Or, les moyens techniques composant un système d'information (ordinateur en particulier) sont conçus pour traiter l'"information". Dans cette acception, le mot "information" évoque un "contenant", un "signifiant".

La conséquence directe de cette distinction fondamentale est le fait que l'utilisation d'outils tel qu'un ordinateur dans un système d'information est une méthode valable lorsque les deux conditions ci-dessous sont respectées:

1) L'information, en tant que chaîne de caractères est une représentation adéquate du renseignement.
2) Le traitement formel que l'on fait subir à cette information réalise bien la combinaison de concepts que l'on désire.

Ainsi la conception d'un système d'information fait-elle généralement appel aux deux moyens; la partie automatisable du système correspondant au traitement de l'information, la partie non automatisable correspondant à des traitements de connaissances.

Le traitement des connaissances est la synthèse faite à partir de l'accumulation des renseignements élémentaires et la définition d'un ensemble de relations entre les différentes éléments; c'est-à-dire, une activité intellectuelle qui comprend des fontions diverses, telles que:

---acquisition de renseignements élémentaires
---classification et organisation en structure générale
---analyse qualitative et quantitative
---confrontation et intégration
---simplification et décomposition par des mécanismes de déduction logique
---synthèse, c'est-à-dire, rassemblement des éléments épars en ensembles cohérents.

MAIS...QUELS SONT LES CONCEPTS DE BASE DE CETTE DUALITE DE L'INFORMATION?

1) ASPECTS FONCTIONNELS : information = "contenu"

Dans cette optique, une "information" apparaît comme un ensemble de trois éléments:

---SUJET : c'est l'objet, l'être, l'entité ou le concept concerné par l'information.
---ATTRIBUT : c'est un élément de la description du sujet; caractère ou propriété de celui-ci.

---VALEUR : c'est le caractère quantitatif ou qualitatif associé a l'attribut.

Exemple 1 :

Lumbricus terrestris	catégorie écologique	anécique
sujet	attribut	valeur

Exemple 2 :

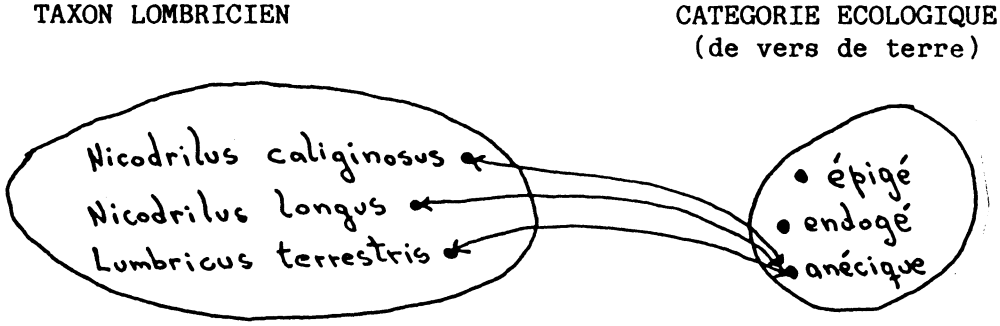
relevé n° ------- 150	ph	6,8
sujet	attribut	valeur

Cet ensemble: "SUJET + ATTRIBUT + VALEUR" est appellé normalement DONNEE.

Si l'on supprime l'un quelconque des trois éléments on perd en pratique presque la totalité de la connaissance apportée par l'information.

Exemple :
"Lumbricus terrestris catégorie écologique" n'apporte aucune connaissance, pas plus que "ph 6,8".

Les sujets et les valeurs des attributs par eux-mêmes sont porteurs de peu d'informations. Le fait de savoir que, par exemple, les entités "Nicodrilus caliginosus", "Nicodrilus longus" et "Lumbricus terrestris" sont des éléments de l'ensemble TAXONS et que "épigés", "endogés" et "anéciques" sont des éléments de l'ensemble CATEGORIES ECOLOGIQUES, indique seulement que deux ensembles d'objets existent. Par contre, si on sait que "Lumbricus terrestris" appartient à la catégorie écologique des "anéciques" ceci devient une information utilisable qui constitue un fait significatif de notre système d'information, parce qu'une association a été construite entre deux ensembles d'objets.



Il est possible qu'un sujet donné puisse être caractérisé par plusieurs propriétés ou étudié selon des points de vue différents; dans ce cas, au sujet seront associés plusieurs attributs et les valeurs correspondantes dans une même information.

Par exemple dans un prélèvement de vers de terre :

INDIVIDU 20	CODE TAXON	A
sujet	1ère attribut	valeur
	CODE STADE	1
	2ème attribut	valeur
	CODE ETAT	2
	3ème attribut	valeur

POIDS	30 mg
4ème attribut	valeur

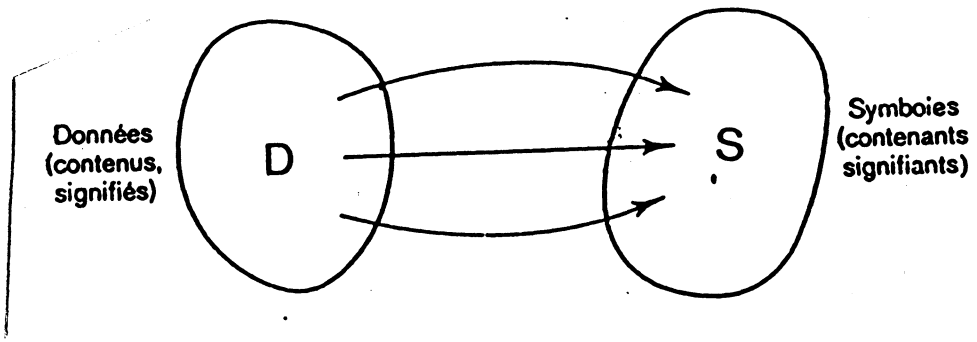
Mais...lorsqu'on veut décrire un système d'information, il faut examiner l'ensemble des objets ou entités qui le composent. Le seul critère qui permet de distinguer une entité ou un objet des autres est le fait de posséder une certaine identité qui le rend discernable des autres. Il faut bien comprendre que le fait de retenir dans le processus de conception d'un système d'information une entité plutôt qu'une autre dépend de l'utilité projetée du système d'information et du niveau d'analyse.

Prenons comme exemple un prélèvement (prélevat) de vers de terre. On peut voir le prélevat dans sa globalité identifié par un numéro de prélèvement ou bien on peut voir chaque individu (vers de terre) trouvé à l'intérieur de ce prélevat comme étant autant d'individualités différentes, car ils se différencient par un numéro, un code taxon, un stade,un état et un poids.

Dans le cadre d'une application considérant l'ensemble des prélèvements chaque prélevat pourra être considéré comme le sujet; alors que dans un problème de détermination à l'intérieur même d'un prélevat, il faudra enregistrer les particularités de chaque individu, et dans ce cas le numéro de l'individu serait le sujet.

2) ASPECTS FORMELS : Information = "contenant"

Dans cette optique, une information doit être représentée sous une forme perceptible. D'une façon plus précise, si "D" est un ensemble de données et "S" un ensemble de symboles, il faut définir une correspondance entre "D" et "S".



Principe de représentation des données

Généralement, l'ensemble des symboles est un ensemble de mots composés à partir d'un alphabet. Cet alphabet comprend l'ensemble des caractères usuels (lettres,chiffres).

Pour éviter les risques de confusion sur le terme "mot", on appellera RUBRIQUE la suite de caractères représentative d'un sujet ou d'un attribut. Une rubrique peut être composée d'un caractère, d'un seul mot (au sens courant) ou de plusieurs mots.

Exemple:

La suite de 3 caractères numériques 1,5,0 (exemple 2) constitue la valeur de la rubrique 150 représentative du sujet "prélèvement N° 150", dans ce cas, le sujet est identifié sous la forme d'un code numérique.

"Lumbricus terrestris" est la valeur d'une rubrique composée de plusieurs mots (suite de caractères alphabétiques avec des blancs intercalés). Dans ce cas le sujet est identifié par son nom "en clair".

"34000" serait la valeur d'une rubrique désignant conventionnellement un département français :l'Hérault.

La correspondance entre l'ensemble des informations et l'ensemble des rubriques ne doit pas être quelconque. A une rubrique doit correspondre au plus une valeur-sujet ou une valeur-attribut pour qu'il n'y ait pas de confusion.

Par exemple, à l'intérieur d'un prélèvement de vers de terre, un seul individu doit être désigné par le N° "20" (valeur de la rubrique sujet de l'exemple antérieur, correspondant à un prélèvement de vers de terre).

En conclusion, et en d'autres termes, le "contenu" est bien renfermé dans un "contenant", mais celui-ci, tel un récipient, ou un verre, peut avoir des apparences diverses, sans pour autant modifier ce qu'il contient.

Si on analyse le problème au niveau d'un ordinateur, afin de ne pas confondre le contenant de l'information et le contenu, à savoir l'information elle-même, on peut désigner le contenant par un symbole et le contenu par ce même symbole placé entre paranthèses.

Exemple:

Soit M une cellule mémoire, le contenu de M est (M). On notera le transfert du contenu de la mémoire M1 dans la cellule mémoire M2 par:

M2 ⟵ (M1) ce qui se lit: le contenu de M1 est transféré dans M2.

STRUCTURE DES DONNEES .... ET FICHIERS

Compte tenu du très grand nombre et de la grande variété de données en écologie, la première tendance est de structurer l'ensemble des informations à traiter sous forme de fichiers.

Un fichier est un ensemble ordonné d'informations, stockées sur un support physique (disque, disquette, bande...) présentant la même structure et pouvant être consultées individuellement, donc, un ensemble d'objets de même nature qui se répètent un certain nombre de fois.

Généralement, un fichier est constitué de données numériques ou alphanumériques nécessaires à l'exécution d'un programme.

COMMENT PEUT-ON IDENTIFIER UN FICHIER?

Tout fichier peut être identifié par;

---un identificateur : son nom
---sa structure logique : structure des enregistrements logiques
---sa structure physique : structure des enregistrements physiques

Exemple :

Fichier CMRXXXXX (observations relatives au milieu)
Fichier CDRXXXXX (observations relatives à la description de la composition floristique ou faunistique).

MAIS ... QU'EST-CE QU'UN ENREGISTREMENT LOGIQUE?

On appelle enregistrement logique (ou article), l'unité d'information donnant lieu au même traitement (quantité d'information traitée par un ordre de lecture ou écriture d'un programme), ou encore ensemble des rubriques (informations) se rapportant à un sujet.

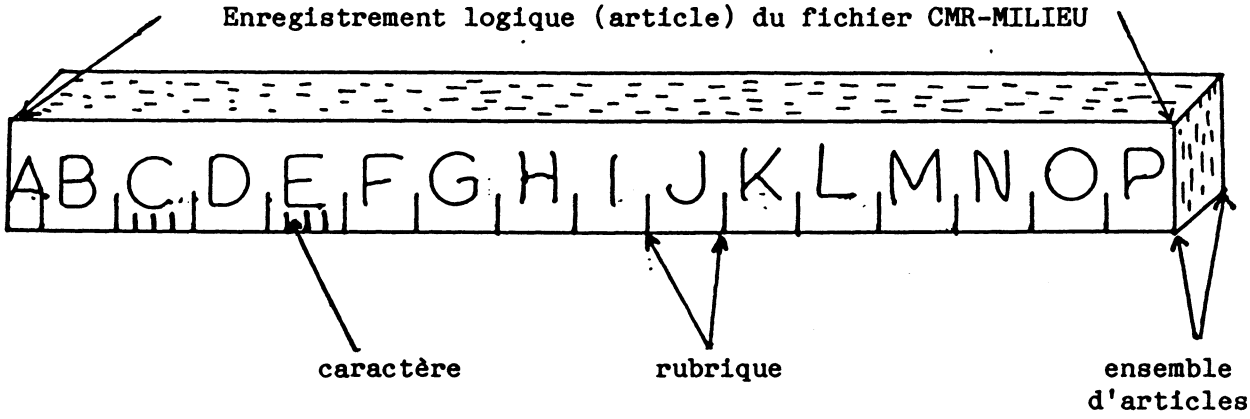
La structure des enregistrements logiques, est une structure externe à l'ordinateur, donc, prévue par l'utilisateur.

Exemple:

CMR-MILIEU (fichier correspondant à l'étude des vers de terre de Belgique). Un enregistrement logique correspond toujours à un prélèvement.

Plan d'enregistrement prévue pour le traitement des observations relatives au milieu:

A : code CMR
B : numéro du prélèvement
C : nombre d'horizons
D : ph horizon supérieur
E : ph horizon inférieur
F : carbone total
G : azote total
H : rapport C/N
I : altitude
J : pente
K : texture observée
L : humidité du sol
M : charge du sol
N : texturé calculée
O : classification phytosociologique
P : humidité du substrat



rubrique = ensemble de caractères
article = ensemble de rubriques
fichier = ensemble ordonné d'articles

Le concept caractère correspond aussi au besoin de coder en machine les différents caractères de l'alphabet (lettres, chiffres, signes). Une longueur de 8 bits, permettant de coder 256 caractères différents, a été presque universellement retenue à tel point que le mot "caractère" désigne souvent un ensemble de 8 bits ou octet (un bit, est la plus petite unité d'information possible qui ne peut prendre que deux valeurs généralement notées 1 et 0).

Alors, le codage consiste à établir une loi de correspondance appelée code entre les informations à représenter et les configurations binaires possibles, de telle sorte qu'à chaque information corresponde une et généralement une seule configuration binaire.

Avec 8 bits, on peut coder jusqu'à 2 puissance 8 = 256 informations différentes. Par exemple, les caractères d'un clavier (machine à écrire). Lorsque l'on appuiera sur une touche, le caractère correspondant sera codé sur 8 bits avant d'être envoyé à l'ordinateur; inversement lorsque l'ordinateur enverra un caractère de 8 bits à l'imprimante, il sera décodé afin de permettre la mise en mouvement du "marteau" correspondant.

Exemple: code ASCII

A ------ 01000001
1 ------ 00110001
, ------ 00101100

QU'EST-CE QU'UN ENREGISTREMENT PHYSIQUE?

Un enregistrement physique correspond à la quantité d'information qui transite en une fois entre la mémoire centrale de l'ordinateur et le support externe (disque,disquette,bande...). La structure physique d'un enregistrement dépend essentiellement des supports technologiques et des traitements prévus.

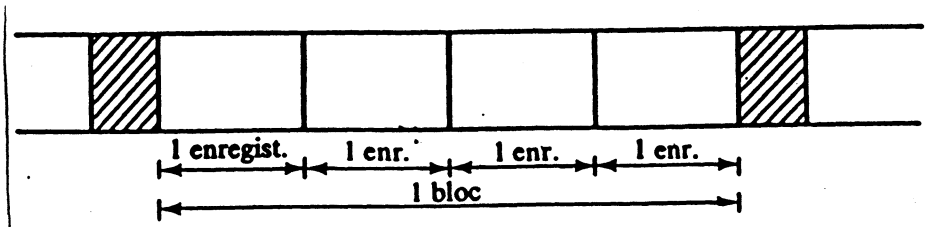
Les traitements prévus déterminent généralement les formats d'enregistrement. Par exemple, quand la longueur des articles est fixe, leur structure interne comportera des rubriques en nombre et longueur fixe ou leur structure interne sera déterminée par un code enregistré dans l'article.

Pour les formats de longueur variable, la longueur de chaque article est déterminée soit par un enregistrement en tête de l'article, soit par un caractère de "fin d'article".

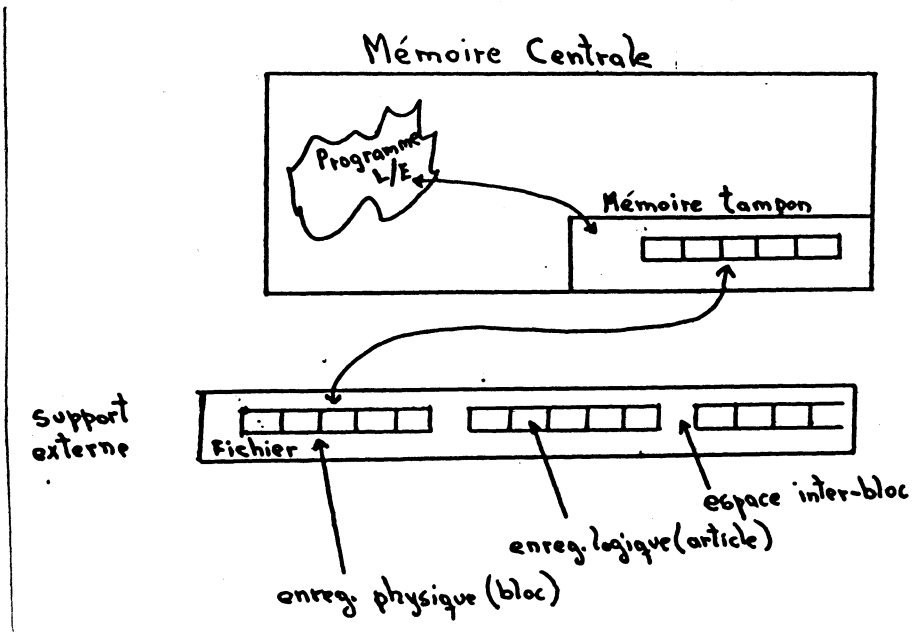
Parfois, la longueur des articles ne coïncident pas toujours avec les contraintes physiques des supports d'enregistrement; dans ce cas, il est utile de grouper plusieurs enregistrements logiques pour constituer un enregistrement physique.

Dans la majorité des cas où l'enregistrement logique est de longueur fixe, l'enregistrement physique doit être un multiple de l'enregistrement logique. Le facteur multiplicatif est appelé facteur de blocage. Donc, l'unité des données physiquement enregistrées est le bloc. Chaque bloc contient N enregistrements logiques, N pouvant d'ailleurs varier d'un bloc à l'autre.

Longueur enreg. logique * facteur de blocage = longueur enreg. physique



1 bloc de 4 enregistrements.



On peut dire aussi, que les articles à l'intérieur d'un fichier ont une structure de "liste" et que les rubriques à l'intérieur des articles ont une structure de "file".

La structure de liste correspond à l'idée de subdivisions successives en sous-ensembles entièrement inclus dans un ensemble plus large, et la structure de file correspond à l'idée d'une collection d'objets considérés successivement, seule une relation d'ordre préside à leur rangement.

Les rubriques d'une donnée doivent être séparées entre elles et séparées de celles de la donnée suivante et leur rangement à l'intérieur d'un article constitue une caractéristique de chaque fichier.

Exemple :

/ a1 b1 c1 d1 / a2 b2 c2 d2 / ..... / an bn cn dn /

séparateurs d'articles — séparateurs de rubriques

Dans l'exemple le "slash" (barre oblique) joue le rôle de séparateur d'article, le signe espace (blanc) joue le rôle de séparateur de rubrique.

Pour éviter toute confusion lors du traitement des fichiers il importe de bien distinguer:

---Ce qui correspond à la désignation du sujet et de l'attribut et qui reste identique pendant tout le traitement (nom de la rubrique).

---Ce qui correspond à la valeur et qui est caractéristique de l'enregistrement en cours de traitement (valeur de la rubrique).

Dans le cas de traitement par ordinateur, le nom de la rubrique est fourni par le programme, les valeurs successives de la rubrique sont obtenues par des instructions de lecture,de calcul,...

Exemple:

nom	valeur
TAXON =	'LUMBRICUS TERRESTRIS'
TAXON =	'NICODRILUS LONGUS'

L'ORGANISATION DES DONNEES SUR SUPPORT EXTERNE

On distingue principalement deux modes d'organisation des données: le mode séquentiel et le mode adressé. Il ne faut pas confondre le concept organisation avec le concept accès, on parle fréquemment d'organisation séquentielle ou directe, alors que ces termes se rapportent uniquement à la manière dont on accède aux données. Il est vrai que cette confusion ne porte pas trop à conséquences, tant que l'on reste dans la conception classique du "fichier informatique", car la structure physique des données sur le support est quasiment identique à la structure logique de ces mêmes données "vues" par le programme.

Dans l'approche d'un problème par les bases de données,il en est tout autrement, car il n'y a plus de lien entre "physique" et "logique", et il importe d'avoir présent à l'esprit que l'accès et l'organisation sont deux choses différentes.

En conclusion, on peut dire que l'organisation est la manière dont les enregistrements sont stockés sur le support, alors que l'accès est la manière dont on accède à ces mêmes enregistrements.

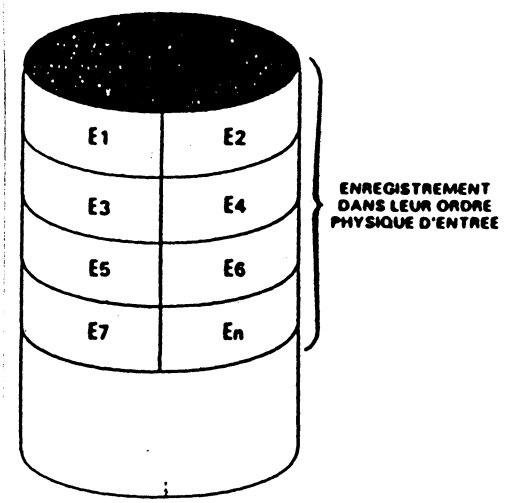
ORGANISATION SEQUENTIELLE ET ACCES SEQUENTIEL

L'organisation séquentielle implique que les enregistrements soient stockés de manière consécutive, c'est-à-dire, chaque donnée est contiguë avec son prédécesseur et son successeur et l'accès à une information donnée ne peut se faire que par consultation de tous les enregistrements situés "avant" et par comparaison avec une clé de référence.

L'organisation séquentielle est la plus simple et la plus couramment utilisée. C'est celle, évidente, des fichiers sur bande magnétique. Les enregistrements sont créés les uns après les autres au fur et à mesure des ordres d'écriture (sur bande ou sur disque).

La relecture se fait également séquentiellement. Ceci implique que pour avoir accès à l'enregistrement de rang "n", il aura fallu, au préalable en lire "n-1".

Le temps de recherche peut être évidement très long, de plus, l'organisation séquentielle est très pénalisante pour les applications en "temps réel" (conversationnel).



ORGANISATION ADRESSEE ET ACCES DIRECT

Le terme "adressé" indique que l'on associe à chaque enregistrement (ensemble d'informations) une notion nouvelle par rapport à l'organisation séquentielle, celle d'adresse, c'est-à-dire, d'emplacement physique sur le support périphérique. Cette organisation est possible sur le support type disque.

Pour qu'une telle organisation puisse exister pour un ensemble de données, il faut qu'il existe un lien, entre l'enregistrement logique et son emplacement physique sur le support. Ce lien s'effectue grâçe à une "clé" ou identificateur logique, associée à un "pointeur" physique indiquant l'adresse de stockage.

La notion de "pointeur" utilisée dans les organisations de type "adressées" apporte justement la matière d'une nouvelle manière de grouper les données.

Le pointeur est généralement une adresse physique revêtant une forme plus au moins différente selon les constructeurs. Par exemple numéro de secteur, numéro de piste, numéro de cylindre ...

Sur une disquette, par exemple, chaque enregistrement logique a une adresse physique définie par le numéro de la piste et la position de l'enregistrement à l'intérieur de cette piste. Il est possible d'accéder à un enregistrement particulier en positionnant directement la tête de lecture/écriture sur la piste où il se trouve sans être obligé de lire tous les enregistrements des pistes précédentes.

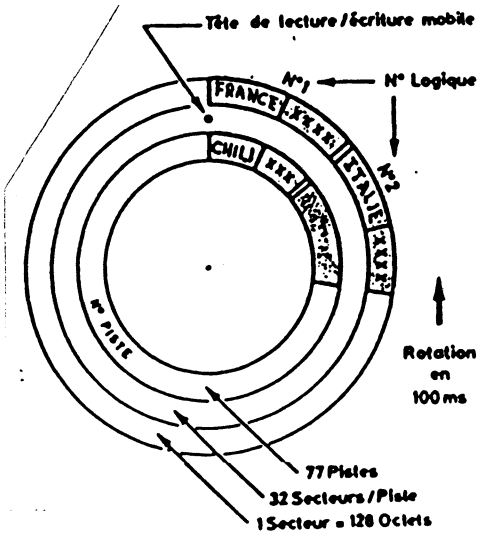
En realité, on n'est pas obligé de connaître les adresses physiques (n° de piste-position de l'enregistrement) on ne connaît qu'un numéro relatif (clé logique) à un début de fichier. A la limite, cette clé n'est autre que le numéro d'ordre de l'enregistrement dans le fichier.

Exemple :

Fichier PAYS , enregistrement N° 6

C'est le logiciel système qui calculera l'adresse physique de cet enregistrement.

Avec cette organisation si on veut rechercher un enregistrement particulier sans avoir à connaître le numéro où celui-ci est rangé, on devra lire séquentiellement tous les enregistrements bien que l'accès soit directe.

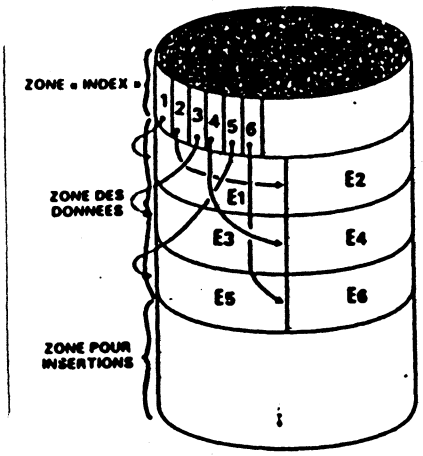


CORRESPONDANCE DANS L'ORGANISATION ADRESSEE ENTRE POINTEUR ET CLE

La correspondance de préférence biunivoque entre clé physique (pointeur) et clé logique peut prendre plusieurs formes:

a) DICTIONNAIRE D'INDEX

C'est un tableau stocké sur un support externe (par exemple: disque), qui relate la correspondance entre clé et pointeur. Pour effectuer une lecture sur un enregistrement déterminé, il suffit de lire en séquence ce dictionnaire et, ayant obtenu l'adresse physique, de se positionner sur le support à l'endroit indiqué.



Mais ..... et si le nombre d'enregistrements figurant sur le fichier est d'un million?

Inconvénient :

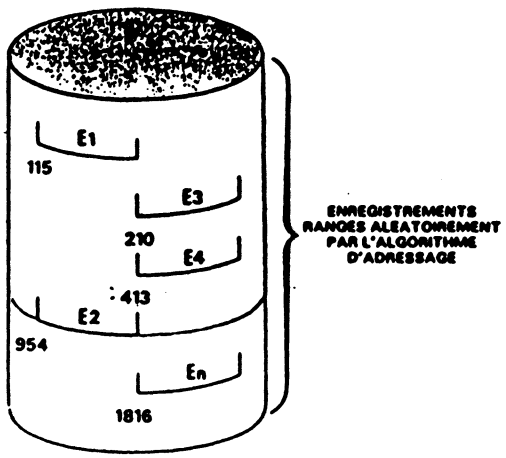
Comme la consultation du dictionnaire se fait d'une façon séquentielle, les temps de réponse risquent de devenir importants quand le nombre d'enregistrements est grand.

Avantage :

En procédant à la lecture séquentielle du dictionnaire, le positionnement est, sans équivoque possible sur l'endroit indiqué.

b) CALCUL DE L'ADRESSE PHYSIQUE

Un algorithme va ainsi être utilisé, qui, à partir d'une clé d'enregistrement, va générer directement l'adresse de stockage correspondante.



..... et si l'adresse calculée est plus grande que la capacité de l'unité de stockage?

Inconvénient :

L'algorithme de calcul des adresses physiques est dépendant du type d'unité de stockage prévue pour le fichier, pour éviter les problèmes de débordement de capacité.

Avantage :

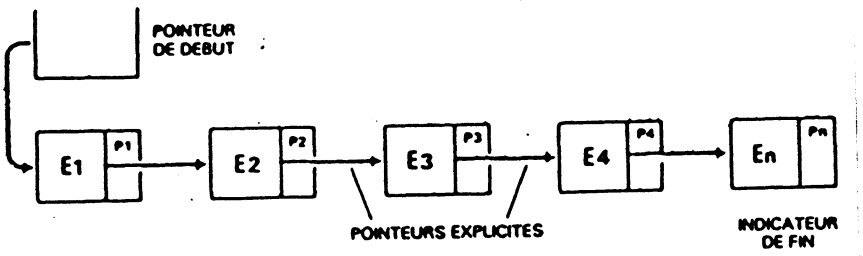
Rapidité des accès puisque la clé physique est directement calculée en mémoire centrale, ayant peu d'accès sur l'unité de stockage.

En conclusion on peut dire, que dans ces deux méthode d'organisation, existe une dépendance entre la structure logique des données et la structure physique du fichier qui les supporte.

Si on veut passer de l'organisation des données par fichier à une organisation Base de Données, il faut, pouvoir dissocier complètement structure logique et structure physique, mais pour cela, il est indispensable que des pointeurs judicieusement choisis soient présents pour rétablir une certaine correspondance.

ORGANISATION PAR LISTE ET ACCES DIRECT

Dans l'organisation par liste, chaque enregistrement contient une ou plusieurs informations spéciales (des pointeurs) qui représentent les adresses d'autres enregistrements, c'est-à-dire, chaque enregistrement peut contenir l'adresse de celui qui le suit et/ou qui le précède "logiquement". Mais les répartitions physiques des enregistrements sur disque seront quelconques.



L'ORGANISATION DES DONNEES EN MEMOIRE

Les données que traite un programme, ne peuvent se présenter en vrac. On doit être en mesure de les nommer, de les adresser et de les manipuler.

Une structure préside donc à leur organisation, qui comporte, comme tout traitement informatique, un double aspect: logique et physique.

Le premier donne lieu à l'organisation abstraite détachée des contraintes matérielles, la seconde à sa realisation concrète sans laquelle une abstraction ne peut prendre corps.

C'est donc sous ces deux visages que nous devons toujours examiner les structures des données.

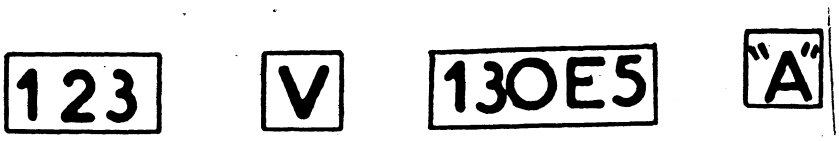
On peut définir trois types de structures des données en mémoire:

a) STRUCTURES STATIQUES : Données scalaires et tableaux, c'est-à-dire, des variables dont l'organisation reste invariante durant le déroulement d'un programme.

- DONNEES SCALAIRES: Ce sont les types de données les plus simples qui soient, leur structure se réduisant à un seul élément. Les données scalaires n'occupent qu'une seule cellule en mémoire et sont considérées comme des types de base.

Ces types de données sont souvent définis dans le langage lui-même: nombre entiers ou réels, caractères, données booléennes.

Exemple:



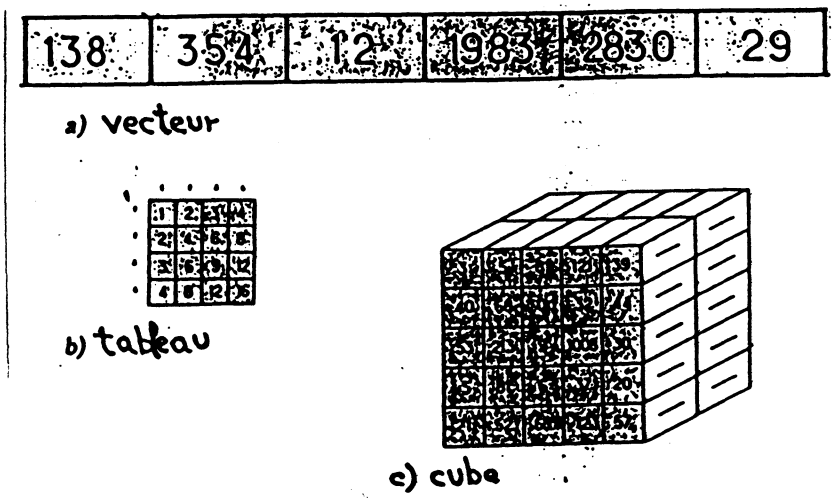
Les opérations permises sur toutes les données scalaires sont: la définition et la création (parfois implicites dans certains langages de programmation), l'affectation et la lecture d'une valeur dans une variable.

Certaines opérations sont limitées à un type particulier. L'addition, la multiplication, la soustraction et la division se trouvent pour tous les types numériques, avec en outre toutes les comparaisons possibles: égalité, relations d'ordre, etc. Les données booléennes autorisent les opérations logiques "et", "ou" et leurs combinaisons, la négation et les tests d'égalité. Les caractères enfin ne permettent que la comparaison.

Chaque donnée occupe une place mémoire: un octet pour les caractères, deux ou plus pour les entiers, quatre au plus pour les réels, et un seul bit suffit à mémoriser les variables booléennes.

- LES TABLEAUX: Sont certainement les structures de données les plus connues. On distingue les vecteurs qui sont des tableaux unidimensionnels, les matrices qui ont deux dimensions et les cubes qui en comportent trois. Bien entendu, il est toujours possible de considérer des tableaux de rang plus élevé.

Exemple:



Tous les éléments d'un tableau sont nécessairement de même type et peuvent être individuellement sélectionnés par l'utilisation d'indices, qui forment les fonctions d'accès aux éléments de la structure.

b) STRUCTURES SEMI-STATIQUES : Piles et files d'attente.

Ces deux notions se réfèrent à des organisations particulières des données en mémoire dans lesquelles l'ordre d'utilisation des informations dépend de l'ordre dans lequel elles ont été introduites.

- La FILE fonctionne selon un principe analogue à celui de la queue d'attente d'un self-service. On a accès à l'information qui a séjourné le plus longtemps selon le principe "premier arrivé, premier servi". Dans ce cas, l'entrée, ou l'écriture, dans la liste s'effectue à l'une des extrémités, alors que la sortie, ou la lecture se fait à l'autre. Par exemple, l'attente de programmes pour disposer d'une imprimante, doit être gérée au moyen d'une file.

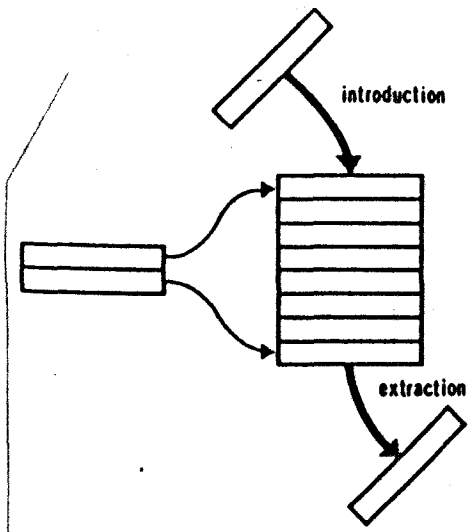
L'implantation physique d'une file se réalise géneralement au moyen d'un tableau et de deux pointeurs. L'un représente l'entrée, l'autre la sortie de la structure. Du fait de l'insertion et de la lecture des éléments par incrémentation des pointeurs, il y a un déplacement continuel de la file vers le sommet du tableau. A cet effet lorsqu'un pointeur arrive au sommet, il est remis à zéro afin de pointer à la base du tableau et être à même de continuer sa tâche. Lorsque le pointeur d'entrée rattrape celui de la sortie, la file est pleine. Si, en revanche, c'est le pointeur de sortie qui rattrape celui d'entrée, alors la file est vide.

- La PILE fonctionne selon un principe analogue à celui illustré par l'exemple des plateaux d'un self-service que l'on prend par le dessus. On a accès à la dernière information rangée selon le principe "dernier entré, premier sorti". Une pile se définit intuitivement comme un empilage d'éléments dont seul le dernier introduit est visible. On peut ajouter ou retirer des éléments mais par un seul bout, le dernier entré devient le premier accessible.

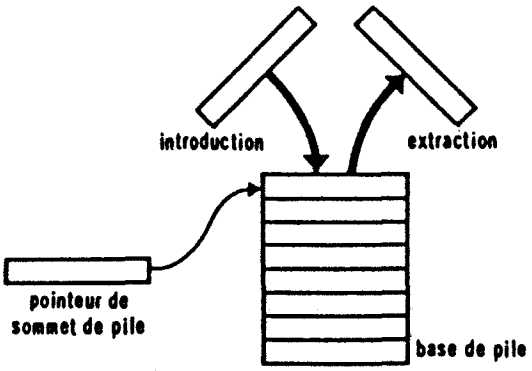
De manière plus rigoureuse, une pile est décrite par un ensemble d'éléments de même type sur lequel sont définies trois opérations: deux fonctions d'accès et un prédicat, c'est-à-dire une fonction de test. Les fonctions d'accès servent à placer ou enlever un élément, le prédicat à déterminer si la pile est vide.

Une pile s'implante en mémoire sous la forme d'un tableau et d'un pointeur qui indique le sommet de la pile. Ajouter un élément revient à le placer à l'endroit où pointe cet indicateur et à l'incrémenter; pour enlever un élément, il suffit d'accomplir l'opération inverse.

Exemple:



Mécanisme de la file d'attente :
premier entré, premier sorti.

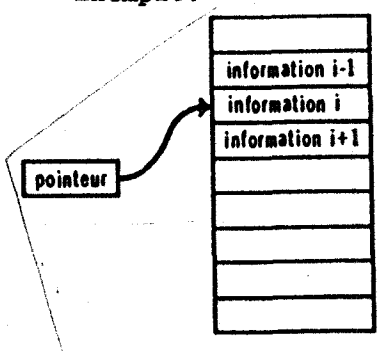


Mécanisme de la pile :
dernier entré, premier sorti.

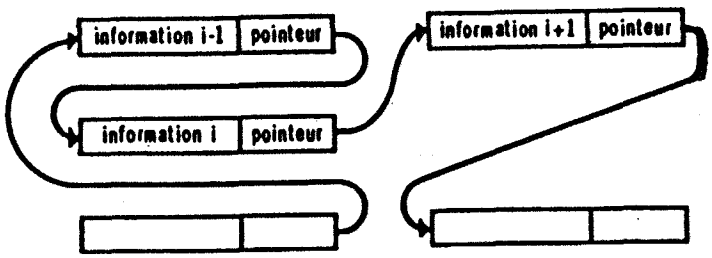
c) STRUCTURES DYNAMIQUES : LES LISTES.

Les structures dynamiques forment la "vie" de l'informatique, son aspect changeant et évolutif. Aucun système, aucun logiciel puissant ne serait possible à l'heure actuelle sans de telles entités. Physiquement, l'élément essentiel est le pointeur. A l'inverse des tableaux dans lesquels les éléments sont sagement alignés les uns à côté des autres, les composantes des structures dynamiques sont disséminées dans l'espace mémoire disponible et reliées entre elles grâce aux pointeurs.

Exemple:



Structure de tableau.



. Structure de liste.

Il existe de nombreuses possibilités d'organisation par listes, dont les p lus connues sont les listes linéaires, circulaires et en arborescence, avec ou sans chaînage dans les deux sens.

LISTES LINEAIRES: Une liste linéaire se décrit logiquement comme une suite ordonnée de taille variable, constituée d'éléments de type défini.

L'implantation d'une liste en mémoire correspond à une structure chaînée, c'est-à-dire à un ensemble d'éléments reliés entre eux par de pointeurs. On pourrait exprimer l'élément d'une liste comme un agrégat de deux champs (un agrégat c'est un regroupement des données de nature différente), "valeur" et "pointeur", le premier contient l'information proprement dite, le second le pointeur sur l'élément suivant de la liste.

Les opérations possibles sur une liste linéaire sont:

---l'accès à un élément particulier de la liste (ceci n'est pas réalisé par l'intermédiaire d'un indice, mais par rapport à un autre élément de la liste);
---l'insertion d'un élément dans la liste;
---la suppression d'un élément de la liste;
---le test pour savoir si la liste est vide ou non.

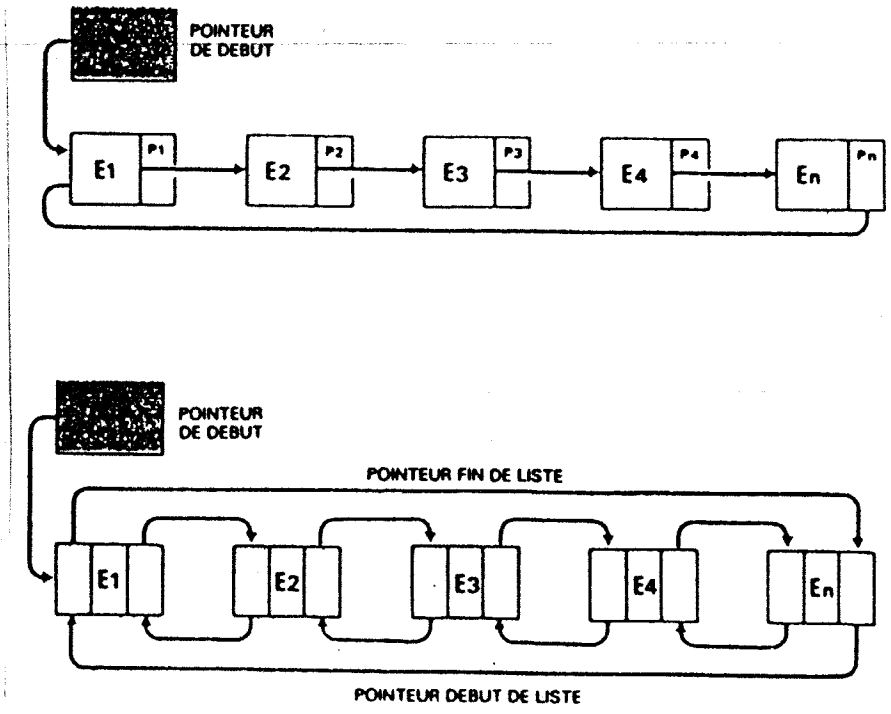
On utilisera donc des listes linéaires chaque fois que l'on aura affaire à un ensemble d'éléments de taille variable (à l'inverse des tableaux qui sont généralement de taille fixe), dans lequel les opérations d'insertion, de suppression et d'accès doivent être accomplies n'importe où.

La structure de liste linéaire peut être améliorée de deux manières différentes:

Par l'emploi d'un double chaînage afin d'obtenir une liste linéaire double, ou par le rebouclage de la fin de liste sur le premier élément, pour constituer une liste circulaire.

Ces nouvelles structures permettent de pallier certains inconvénients de la structure linéaire simple: lecture des éléments dans les deux sens (listes linéaires doubles), diminution de l'importance accordée au premier élément de la liste (listes circulaires).

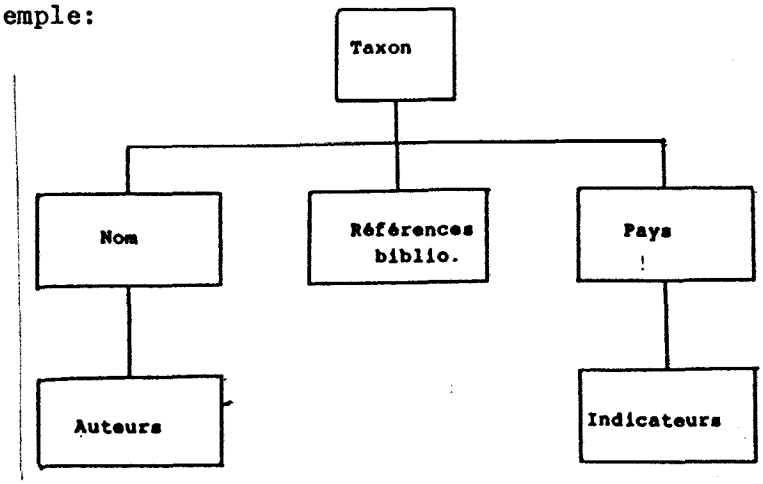
Exemple:



LISTES EN ARBORESCENCES: C'est un ensemble d'éléments organisés de façon hiérarchique. Les "arbres" en informatique poussent vers le bas. Aussi dit-on que la racine de l'arborescence se trouve à son sommet, les branches pendant vers le bas, les feuilles étant les éléments terminaux, c'est-à-dire les plus basses de l'arbre.

Une importante caractéristique des arbres est de pouvoir être "parcourus", c'est-à-dire qu'il est possible de se déplacer le long de cette arborescence dans un certain ordre et de traiter les valeurs des noeuds au fur et à mesure de ce parcours.

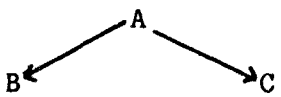
Exemple:



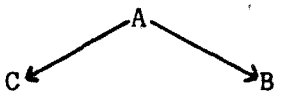
Les arborescences quelconques n'ont pas de représentation physique directement appropriée. Afin de pouvoir implanter cette structure, il faut analyser une arborescence d'un type particulier: l'arbre binaire, qui se représente directement en machine; de plus, tout arbre peut se ramener à un arbre binaire.

Un arbre binaire est un arbre dont chaque noeud ne possède que deux branches, et pour lequel on fait une différence entre le "fils-gauche" et le "fils-droit".

Autrement dit un arbre de la forme:



sera différent de



Les opérations possibles sur un arbre binaire sont:

---Accès, qui se différencie en trois fonctions:
1.- Racine, qui lit la racine d'un arbre
2.- Droite, qui lit la branche droite
3.- Gauche, qui donne accès à la branche gauche.

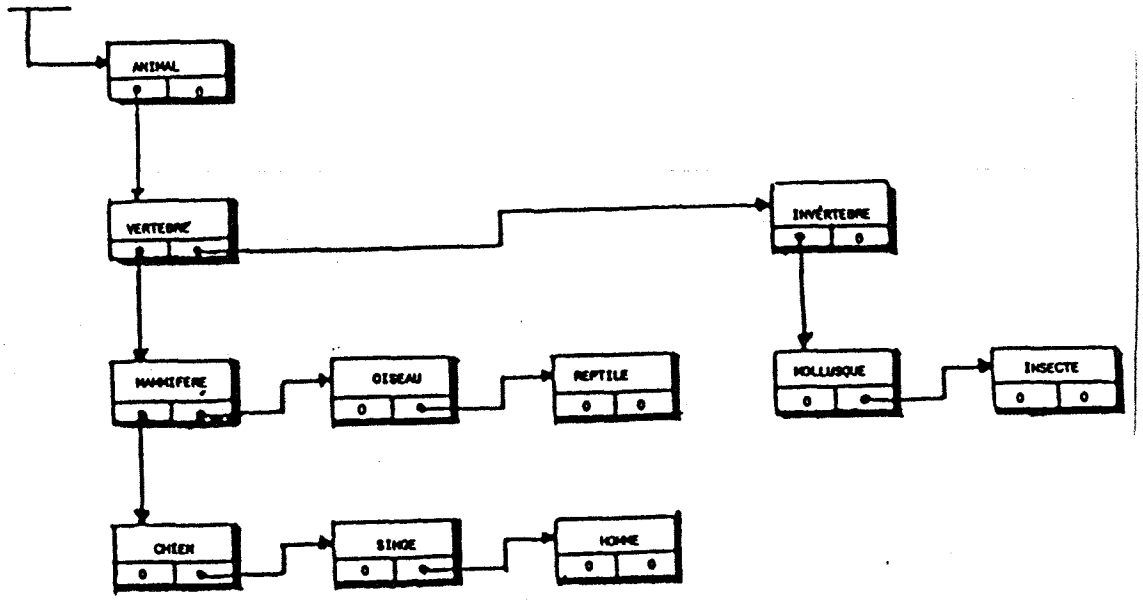
---Construction, création d'un arbre binaire à partir de deux branches et d'une racine.

---Test, fonction vide qui détermine si une branche est vide ou non.

Il est possible, de transformer une arborescence quelconque en un arbre binaire, par l'intermédiaire d'une transformation "canonique", c'est-à-dire qui marche dans tous les cas.

Cette transformation revient à faire pointer les fils gauche vers le premier des enfants, et le fils droit vers les éléments de même niveau que la racine. Le fils droit se transforme ainsi en frère cadet.

Exemple:



Représentation canonique de l'arborescence animalière sous la forme d'un arbre binaire

Il existe d'autres transformations possibles d'arborescences en arbre binaire, et de nombreuses structures complexes se réduisent à un arbre binaire.

DES FICHIERS ..... AUX BASES DE DONNEES

LES PRINCIPES DE BASES DE DONNEES

Historiquement, chaque nouvelle application engendrait ses propres fichiers et ses propres programmes. La création d'une base de données va à l'encontre de cette façon de faire: elle rend possible la centralisation, la coordination, l'intégration et la diffusion de l'information archivée. Actuellement les progrès technologiques permettent de stocker des masses de données de plus en plus grandes pour un coût de plus en plus faible.

Les fichiers entre eux possèdent la plupart du temps des éléments communs, des associations qui ne sont pas exploitées du fait qu'ils sont utilisés isolément et indépendamment les uns des autres.

Par ailleurs, il paraît évident de dire que pour chercher une information, il est nécessaire de savoir où elle se trouve! En fait, comme on vient de le souligner, en informatique traditionnelle, on est souvent amené à poser la question à cause des nombreux fichiers mis en oeuvre.

C'est un problème qui ne devrait plus se poser en raisonnant "base de données", car la "non-redondance" de l'information est l'un des principes de base.

Mais ..... en général, qu'est-ce qu'une base de données?

Une base de données est tout simplement un ensemble de données structurées sous forme de listes et enregistrées sur des supports accessibles par l'ordinateur.

La notion de liste permet l'utilisation des seules informations nécessaires; l'utilisation des seules informations nécessaires est l'un des points les plus importants d'application des données.

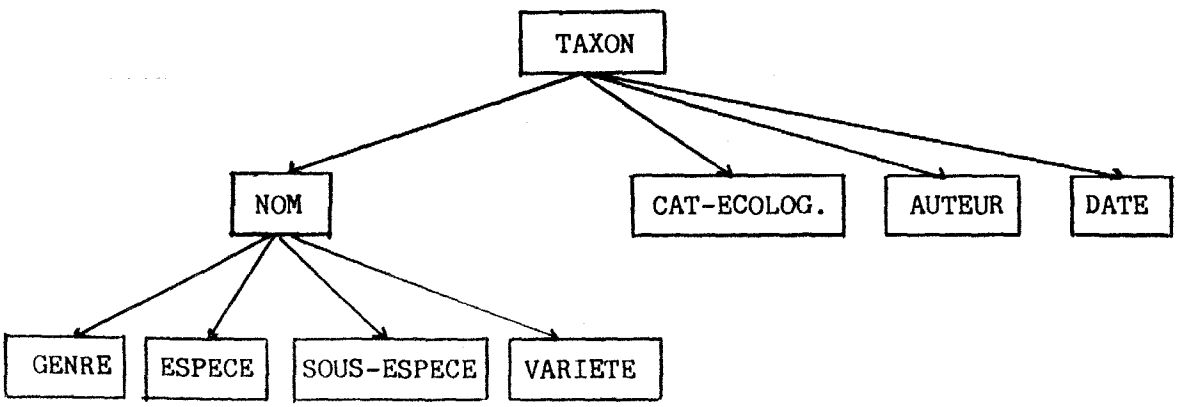
En fait, que peut nous apporter la notion de liste par rapport à celle d'enregistrement?

Dans une structure d'enregistrement on trouve habituellement regroupées l'ensemble des informations utiles, dont la longueur est la somme de celles des zones élémentaires

Par exemple, l'enregistrement suivant:

GENRE	ESPECE	SOUS-ESPECE	VARIETE	CAT-ECOLOG.	AUTEUR	DATE

fait apparaître SEPT zones élémentaires, dont on peut faire ressortir la structure de liste en arborescence de la manière suivante:



On constate que la décomposition hiérarchique de cet enregistrement fait ressortir DEUX zones "fictives": TAXON et NOM, qui n'ont aucune realité physique (elles ne figurent pas dans l'enregistrement), mais qui vont servir à établir cette hiérarchie.

On peut d'ailleurs noter que si la zone "NOM" est facultative, c'est-à-dire qu'elle ne se justifie que si l'on veut accéder globalement aux quatre informations élémentaires: Genre, Espèce, Sous-espèce et Varieté; en revanche, "TAXON" n'est pas facultative, car elle correspond au nom de l'enregistrement dans sa totalité.

Quels sont les inconvénients de la structure par enregistrement?

Pour une solution informatique classique, on peut dire aucun. Car la "chaîne de traitement" étant conçue de telle façon qu'elle considère comme un tout les ensembles de données (fichiers) qu'elle traite, il n'y a pas lieu de se poser des questions quant à des possibilités nouvelles d'accès aux informations.

..... On est encore fort loin du concept de "base de données" !

Supposons que, dans notre exemple, nous désirions connaître la totalité des taxons présents dans chaque catégorie écologique. Organisée par enregistrements, notre application va imposer une lecture séquentielle du fichier. Mais, alors que seule la zone élémentaire CAT-ECOLOG. est utile, il faudra quand même amener en mémoire les zones GENRE, ESPECE, SOUS-ESPECE, VARIETE, AUTEUR et DATE qui sont sans intérêt dans ce cas.

..... Alors, la notion de base de données va résoudre ce problème (entre autres). En effet, si nous reprenons notre structure hiérarchique précédente, chacun des enregistrements a été découpé en ses zones élémentaires, appelées "segments", on pourra explorer en séquence toutes les occurrences du segment CAT-ECOLOG. sans se préocupper des autres.

Si on veut lister tous les taxons du genre LUMBRICUS on lira et testera toutes les occurrences du segment NOM.

Grâce à la structure en base de données, l'ordinateur va nous fournir les informations dont nous avons besoin et non celles qu'il nous impose!

Il est très important de bien comprendre la différence fondamentale entre la notion de fichier et la notion de base de données. En effet, vu de loin, on peut toujours se dire qu'il y a des milliers de caractères stockés à la suite sur des disques et que l'accès se fait en séquence ou directement. Il paraît bien difficile d'inventer autre chose. Pourtant, en regardant les choses de plus près, on s'aperçoit que la base de données n'est pas un ensemble de fichiers mais un ensemble d'informations. Il y a donc une approche "logique" bien plus importante que les réalités physiques.

ATTENTION! ..... on va entrer dans des considération de conception de l'informatique.

Que fait un système classique?.....il traite de l'information: on crée des fichiers, on entre des données, on les lit, on fait quelques manipulations et opérations et on écrit le résultat sur un autre fichier, c'est-à-dire, cette informatique-là se fait plaisir en manipulant des quantités de fichiers toujours plus grande.

Ne ferait-on pas mieux de moins "traiter" cette information, mais d'en faire profiter un plus grand nombre et dans de meilleures conditions?

L'approche "bases de données" est sans doute l'orientation qui peut le plus apporter à l'informatique, engluée actuellement dans ses fichiers monstrueux et inexploitables car la plupart du temps incohérents.

Combien d'applications dorment actuellement sur des bandes, disques ou tout simplement sur des "listings" au fond d'un tiroir?..... Coordoner les efforts, accumuler des données non redondantes, faire circuler l'information: voilà l'un des buts principal de la constitution des bases de données.

NOTION DE SYSTEME DE GESTION DES BASES DE DONNEES

Le logiciel qui permet à un utilisateur d'inter-agir avec une base de données s'appele SYSTEME DE GESTION DE BASE DE DONNEES (S G B D). Il permet principalement, d'organiser les données sur les supports périphériques et fournit les procédures de recherche et de sélection de ces mêmes données.

Les fonctions que doit assurer un SGBD sont:

--- Mettre à la disposition de l'utilisateur un outil pour décrire l'ensemble des données qui seront stockées dans la base de données.

--- Offrir à l'utilisateur un outil d'interaction avec la base de données sous forme d'un dialogue pour rechercher, sélectionner et modifier des données.

--- Offrir à l'utilisateur la possibilité de définir des règles qui permettent de maintenir l'intégrité de la base de données. Ces règles sont appelées des contraintes d'intégrité. Elles correspondent à des propriétés qui devront toujours être vérifiées dans la base quelles que soient les valeurs enregistrées.

--- Offrir des mécanismes permettant de vérifier les droits d'accès des utilisateurs.

--- Offrir des mécanismes qui permettent de détecter les cas où il y aurait conflit d'accès et de les traiter correctement. Exemple: deux utilisateurs accèdent aux mêmes informations de la base de données en même temps.

--- Offrir la sécurité de fonctionnement, en cas d'incident (matériel ou logiciel) le système doit permettre la prise de point de contrôle pour remettre la base de données dans un état satisfaisant.

NOTION DE MODELE DE BASE DE DONNEES

L'utilisation de tout système de base de données nécessite dans une première phase une description de l'application envisagée en termes de données à stocker, et de traitement à effectuer. Cette description ne peut être faite que si on dispose d'un modèle de base de données pour comprendre et interpréter l'application.

Tous les logiciels de gestion de base de données utilisent implicitement un type de modèle. Les principaux modèles qui sont utilisés au travers des différents systèmes sont:

--- les modèles hiérarchiques
--- les modèles réseaux
--- les modèles relationnels

L'évolution historique s'est faite progressivement des modèles hiérarchique et réseau vers le modèle relationnel.

NOTION DE LIEN

Un lien peut se définir comme la représentation d'une association (relation entre deux ensembles d'entités, non forcément distincts). L'association est une perception abstraite de la réalité alors que le lien va être sa matérialisation. Le lien offre un mécanisme d'accès qui, à partir d'un élément d'un ensemble permet d'accéder aux éléments d'autres ensembles qui lui sont reliés.

Dans la realité, suivant le SGBD, il existe des limitations qui induisent des liens de différentes natures dus à la caractérisation de l'association qui existe entre les différents types d'informations.

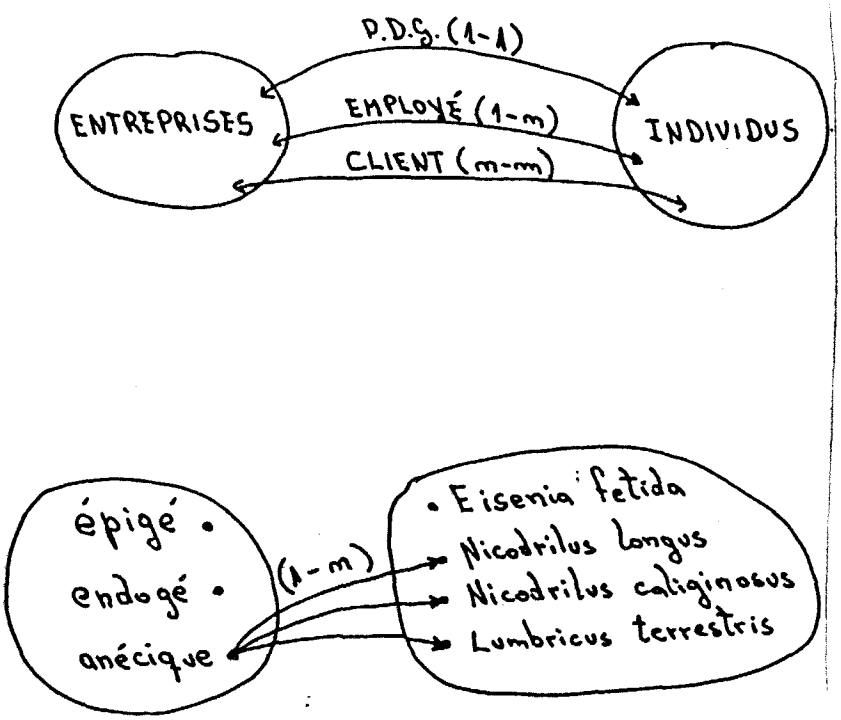
Quels sont les types de liaisons entre les informations?

a)La liaison 1-1: permet de relier deux types d'informations, une à chaque bout, parfaitement symétriques. On parle de liaison binaire.

b)La liaison 1-n: dans cette liaison à une extremité on trouve une seule information et à l'autre extremité il y en a "n". Ce n'est pas une liaison symétrique, c'est une liaison orientée. Cette liaison est la liaison de type hiérarchique.

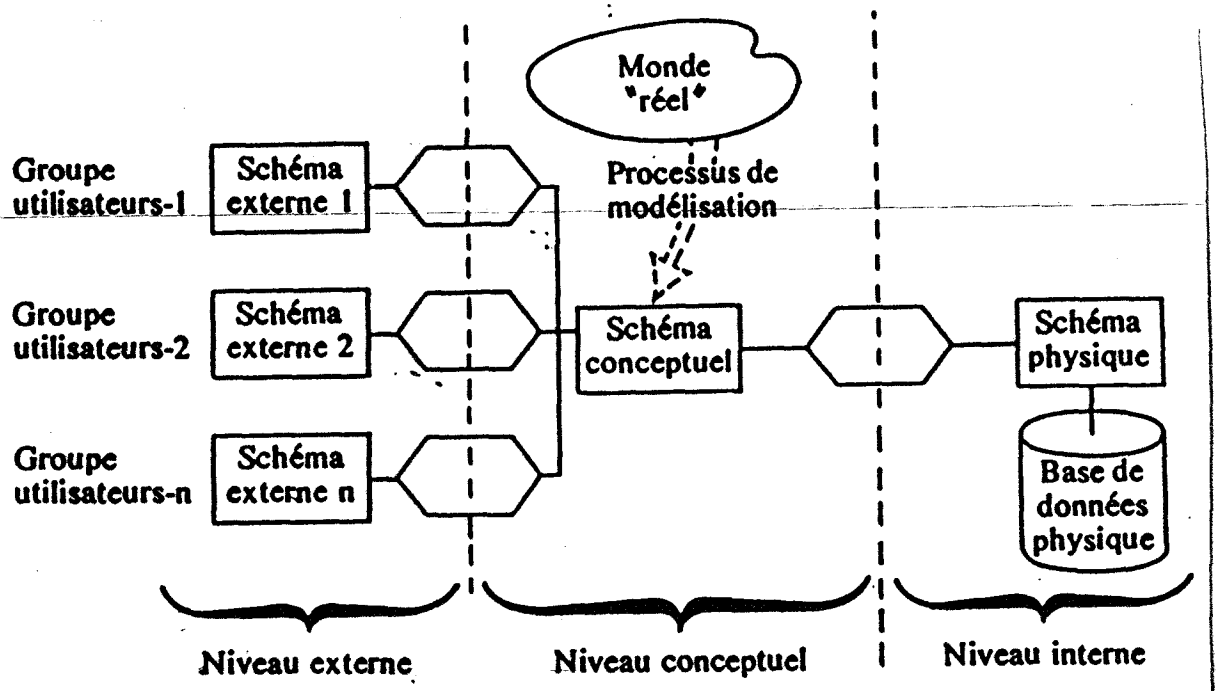
c)La liaison n-m: constitue une réalisation de liens où chaque élément d'un ensemble R peut être associé à m éléments d'un ensemble S, et chaque élément de l'ensemble S peut être aussi associé à n éléments de l'ensemble R. On peut représenter cette liaison d'une manière symétrique.

Exemple:



LES DIFFERENTS NIVEAUX DE REPRESENTATION D'UNE BASE DE DONNEES

Il est couramment admis aujourd'hui qu'il existe trois niveaux de représentation d'une base de données: le niveau interne avec le schéma physique, le niveau conceptuel avec le schéma conceptuel et le niveau externe avec les schémas externes.



Les différents niveaux de représentation d'une base de données.

LE NIVEAU INTERNE

Le schéma physique a pour but de spécifier comment les données seront stockées sur les organes périphériques (disque, bande,...) de l'ordinateur.

Il s'intéresse aux questions suivantes:

---Quel est le volume de données à stocker?
---Quelles sont les unités nécessaires (ordinateur, mémoires, terminaux)?
---Comment les données seront-elles réparties sur les unités de disques?
---Comment les données seront-elles utilisées? Interrogation, exploitation, ou les deux? Laquelle de ces activités va utiliser le plus de ressources?
---Quel sera le nombre d'utilisateurs? Combien d'entre eux utiliseront le SGBD simultanément?

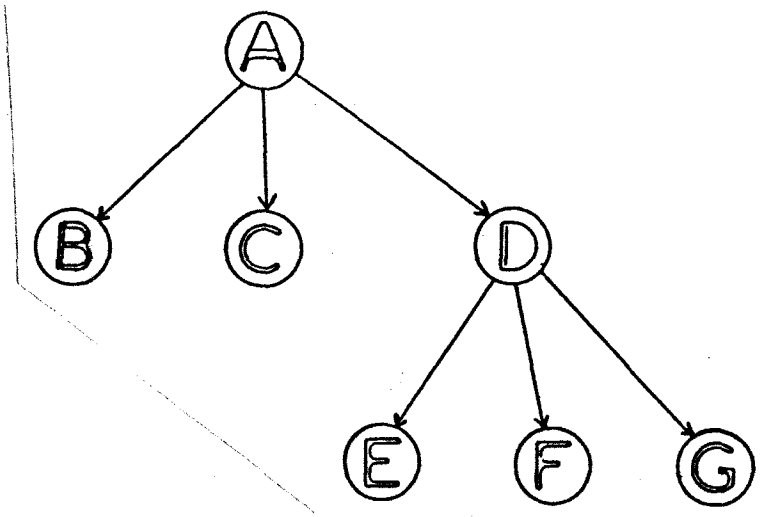
A des questions de ce type il faut répondre en terme de configuration initiale et en terme de besoins anticipés pour l'avenir proche.

LE NIVEAU CONCEPTUEL

Le schéma conceptuel est la partie fondamentale dans l'architecture d'un système de base de données. Il a pour but de décrire en termes abstraits mais fidèles une certaine réalité d'une organisation et de ses processus de gestion qui nécessitent la mise en oeuvre d'une base de données. C'est un processus de modélisation où les objets du "monde réel" sont classés en catégories et désignés par des noms à l'aide des termes et des expressions permises par le langage de définition de données fourni par le SGBD et en respectant un type de modèle de données.

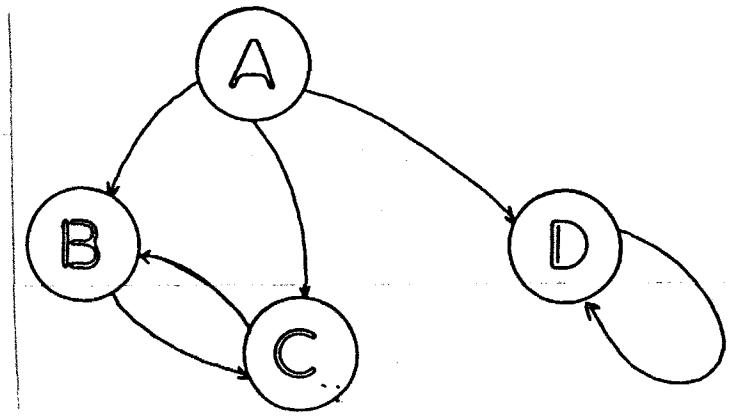
A l'aide du modèle hiérarchique, le schéma conceptuel peut être visualisé sous forme de graphe arborescent dont les noeuds correspondent aux classes d'objets et les arcs entre deux noeuds aux associations. Un tel graphe possède un noeud racine, et les autres noeuds sont des fils, petits-fils, etc, de ce noeud racine.

Exemple:



A l'aide du modèle réseau, le schéma conceptuel pourra être visualisé sous forme d'un graphe général où les noeuds sont des classes d'objets et un arc entre deux noeuds représente une association. A la différence du modèle hiérarchique, la représentation graphique obtenue ne comporte aucune limitation.

Exemple:



A l'aide du modèle relationnel, le schéma conceptuel pourra être représenté par des associations d'ensembles d'objets fondé sur la notion mathématique de relation et visualisé sous la forme de tableaux.

Exemple:

R (A, B, C, D)

R

A	B	C	D

LE NIVEAU EXTERNE

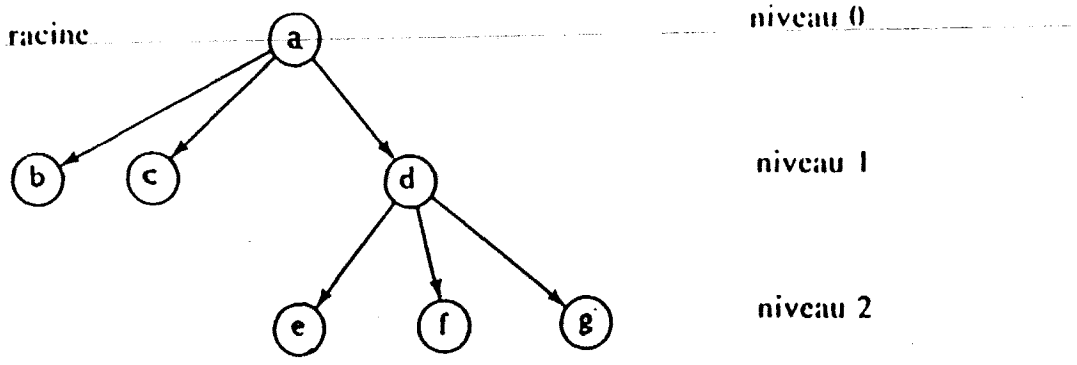
Ce niveau correspond à la vision de tout ou partie du schéma conceptuel par un utilisateur ou groupe d'utilisateurs concerné par une application. Il s'agit donc de décrire, à l'aide d'un schéma externe (parfois appelé vue), la façon dont seront perçues les données par un programme d'application.

LE MODELE HIERARCHIQUE

Ce modèle est capable de créer et de gérer des bases de données dont les relations entre les divers éléments logiques sont du type communément appelé "un à n", ces éléments sont reliés entre eux pour constituer une arborescence valuée.

Une arborescence est un graphe dans lequel un noeud peut recevoir au plus une flèche et d'où il peut partir un nombre quelconque de flèches, donc une arborescence est un ensemble de noeuds reliés par des flèches, appelées "branches" dans la terminologie du modèle hiérarchisé, et tel qu'un noeud particulier unique de l'arborescence est appelé la "racine" car il ne reçoit aucune flèche

Exemple:



Une arborescence.

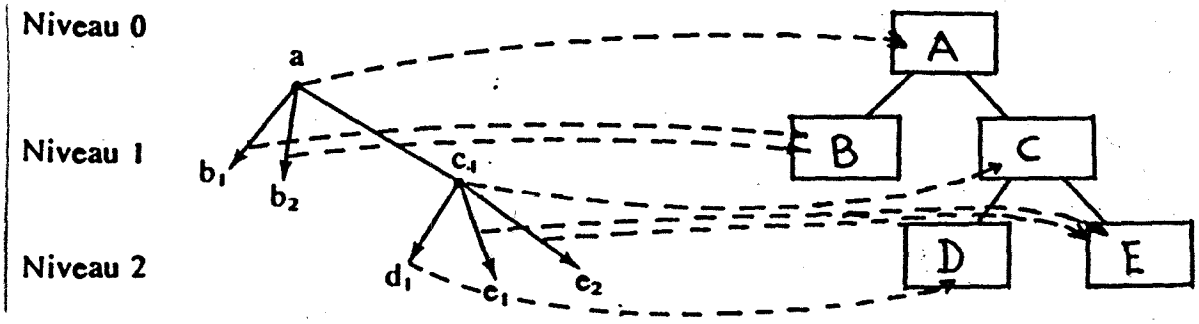
On appelle niveau d'un noeud la distance d'un noeud a la racine, par définition la racine est toujours au niveau 0. On appelle fils d'un noeud l'ensemble des noeuds à distance 1 que l'on obtient en parcourant les branches partant du noeud. On appelle père d'un noeud la notion inverse de celle de fils. Dans une arborescence, tout noeud n'a qu'un père et peut avoir plusieurs fils.

Une arborescence valuée est une arborescence dont les noeuds sont valués par des éléments logiques de nature différente. Elle est définie à partir d'un schéma hiérarchique et doit satisfaire aux contraintes suivantes:

a) La racine de l'arborescence valuée est du type de la racine du schéma hiérarchique.

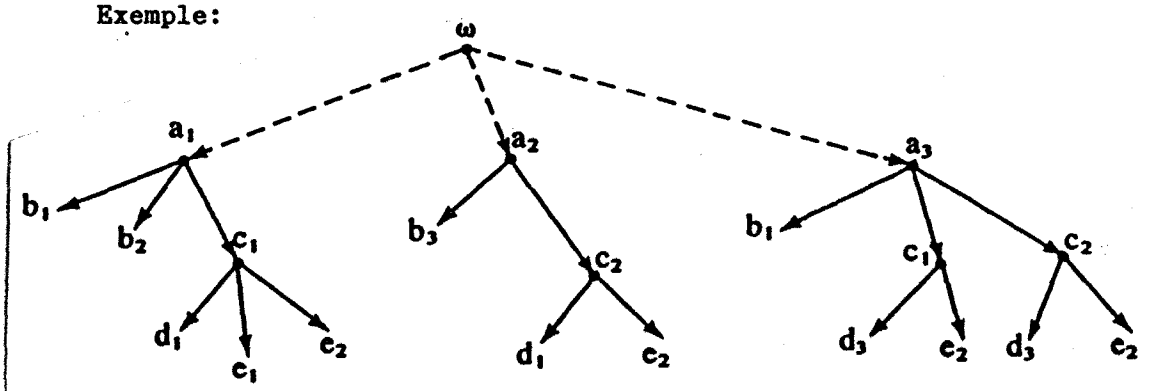
b) Si un noeud c1 de type C est le fils d'un noeud a1 de type A, alors C est aussi le fils de A dans le schéma hiérarchique.

Exemple:



Dans la plupart des cas pratiques, il faut plusieurs arborescences valuées associées à un même schéma hiérarchique pour représenter le contenu de la base de données. On dit alors que l'ensemble des arborescences valuées constitue une forêt.

Exemple:



Une forêt d'arborescences valuées.

Dans l'exemple précédent, des noeuds distincts peuvent être valués avec la même valeur, dans l'arborescence de racine dont la valuation est a3, les valeurs c1 et c2 apparaissent déjà dans les autres arborescences. Ceci signifie que l'information est redondante.

En résumé, on peut dire que une base de données hiérarchisée est constituée par l'ensemble des arborescences valuées qui font référence à un même schéma hiérarchique.

L'implantation physique d'une base de données hiérarchisée correspond fondamentalement à une structure de liste où chaque élément de la liste contient la valuation d'un noeud et plusieurs types de pointeurs, par exemple:

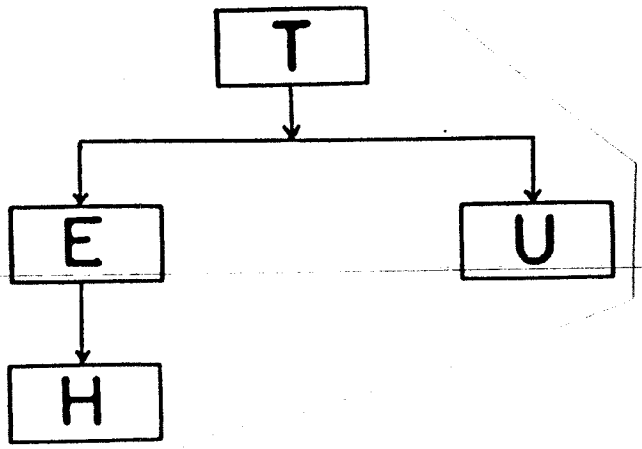
-un pointeur père qui, pour chaque noeud, pointe vers le père de ce noeud.

-un pointeur fils pointant vers le premier fils gauche d'un noeud. On peut même créer autant de pointeurs fils différents qu'il y a de liens hiérarchiques différents dans le schéma hiérarchique.

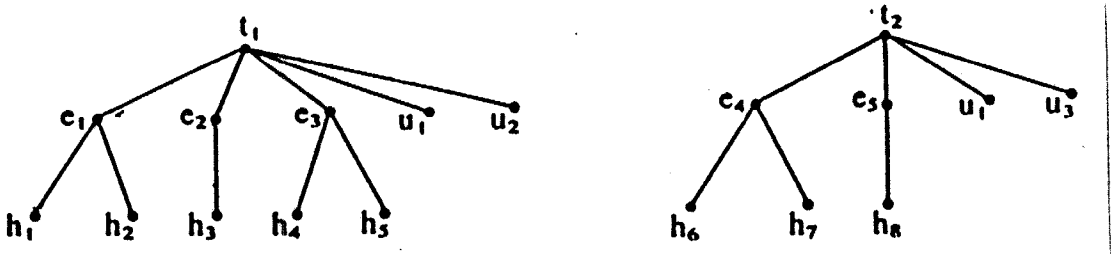
-un pointeur frère pointant vers le noeud suivant de même type.

Exemple:

a) schéma hiérarchique



b) arborescences valuées



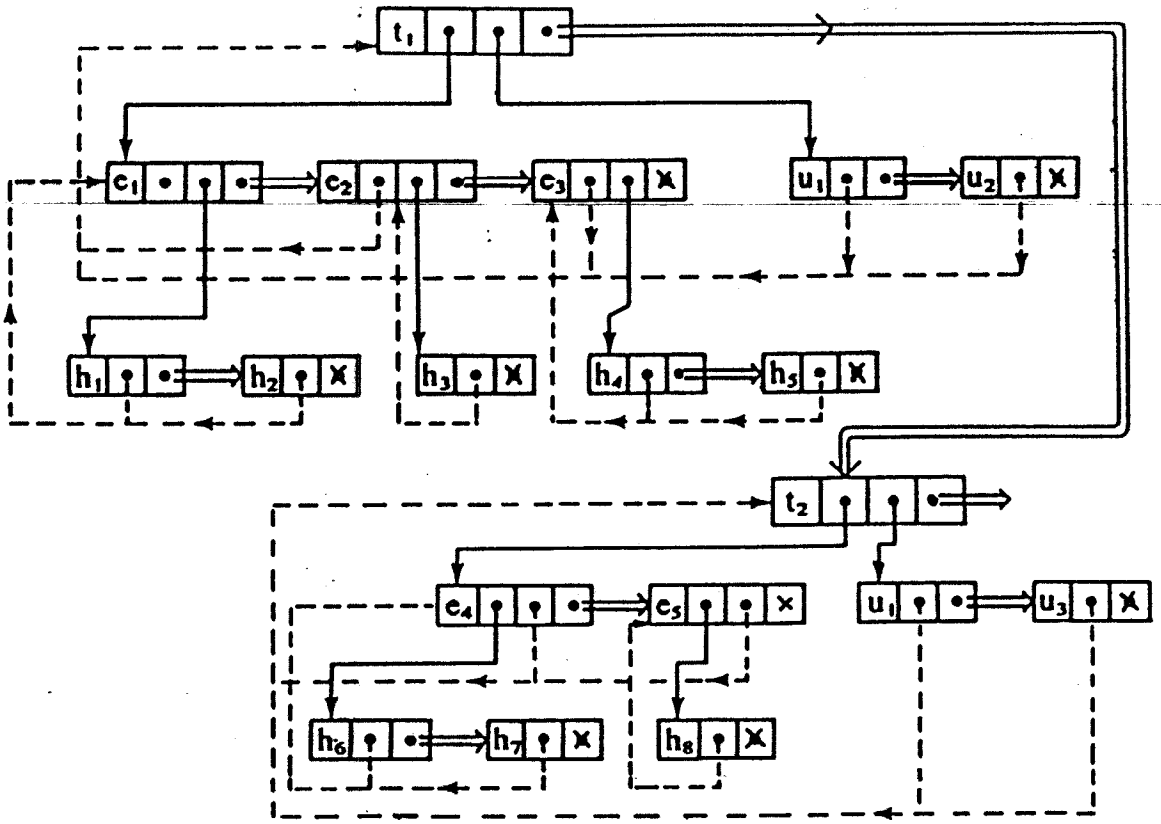
c) ensemble de listes

t1, e1, h1, h2, e2, h3, e3, h4, h5, u1, u2

t2, e4, h6, h7, e5, h8, u1, u3

d) implantation physique

fig. 27



Légende :

pointeur fils
pointeur frère
pointeur père
pointeur nul

LE MODELE RESEAU

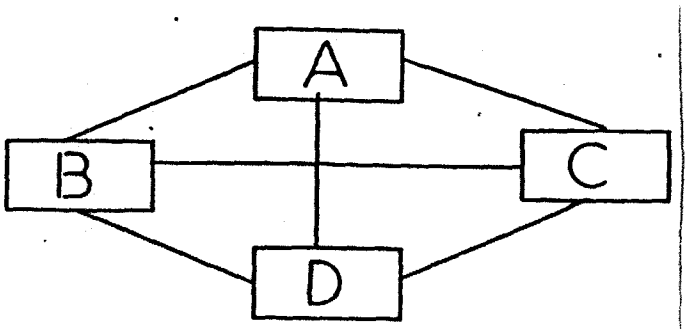
La structure hiérarchique pure ne permet pas en effet de répondre à tous les types d'interrogations, car le principe de base est la dépendance des unités d'information entre elles de type "1-n".

Dans la structure réseau il est possible de définir des liens pour lesquels cette dépendance n'est pas évidente à priori. Dans ce modèle, il n'y a pas la limitation telle qu'une information doit dépendre seulement d'une autre. Ici une information va pouvoir être reliée à un nombre quelconque d'informations et par des liaisons qui ne sont pas forcément orientées. Dans le modèle réseau peut exister une liaison entre deux informations parfaitement symétriques.

Par exemple, on peut aussi bien partir de "A" pour aller vers "B", que de "B" pour aller vers "A" et le nombre de liaisons ainsi établies est quelconque; c'est-à-dire, qu'on peut établir entre "B" et "A" non pas une seule liaison, mais plusieurs.

Dans ce modèle il n'y a plus de points d'entrée privilegiés, on peut entrer par n'importe quelle information, aussi bien par "A", par "B", par "C" ou par "D".

Exemple:



Dans le modèle réseau, il va falloir nommer non seulement les informations, mais aussi les liaisons et au moment de l'exploitation on va indiquer chaque fois par quelle liaison on passe d'une information à une autre.

On voit bien que, dans ce modèle il n'existe pas à proprement parler de lien de subordination entre les éléments, puisque ces données sont logiquement au même niveau hiérarchique.

Dans un système réseau, il est possible de répondre à tout un ensemble d'interrogations mettant en jeu des données sans liens de dépendance les unes par rapport aux autres. Mais la complexité d'une telle base va croître très vite (les temps d'accès aussi) en fonction du nombre de liens supplémentaires (pointeurs) et des chemins d'accès à utiliser.

DES MODELES HIERARCHIQUE ET RESEAU ..... AU MODELE RELATIONNEL

Tout processus de conception doit commencer par une analyse de la situation existante tant sur le plan des informations que sur celui des organisations. Les résultats de cette phase initiale donnent naissance à un schéma conceptuel, c'est-à-dire, à une modélisation informatique du système d'informations.

LES PROBLEMES DES MODELES HIERARCHIQUE ET RESEAU

---La souplesse de faire créer par le système tous les pointeurs desirés est limitée par la nécessité de définir au moment de la création de la base, l'ensemble des pointeurs et chemins d'accès voulus.

---Toute modification ultérieure implique généralement une refonte de la base et des programmes d'exploitation.

---La mise en oeuvre de ces systèmes est complexe et riche en embûches de toutes sortes et de problèmes de dernière minute, même si l'analyse a été bien menée.

---Difficultés pour l'utilisateur sur:
a)la conception et la mise sur pied de la base;
b)l'utilisation et l'évolution de la base.
Par exemple:
+compréhension du système de gestion de la base de données;
+problème de la définition de son application;
+taille de la documentation technique;
+complexité des langages de commande du SGBD
+ennuis rencontrés lors de la mise en route d'une application;
+toute modification de la structure de la base entraîne automatiquement un "arrêt" temporaire de l'activité de la base en question.

---Les liens physiques doivent être spécifiés au moment de la construction de la base, donc toute modification ultérieure n'est pas permise par le SGBD sans reconstruction de la base.

LES AVANTAGES DU MODELE RELATIONNEL

Parmi les modèles de données proposés pour décrire un schéma conceptuel, les modèles relationnels, ont fait progresser de manière significative la conception et la spécification de ces schémas.

L'approche relationnelle apporte de très nombreux avantages aux usagers d'un système de base de données.

Par exemple:

---La description simplifiée des données. Système plus proche du raisonnement humain, que celui hiérarchique ou réseau.

---L'indépendance entre les données et les programmes. Toute adjonction ou suppression de données se fera par modification des relations sans la nécessité de toucher les programmes d'applications.

---Le SGBD agit "par déduction" à partir de l'analyse de la question. Il permet de répondre à toute question formulée correctement faisant appel à une logique déductive, le contenu des questions va déterminer quels sont les chemins d'accès à utiliser et les liens à établir.

---L'indépendance des informations par rapport aux structures physiques de stockage. Les liens ou pointeurs ne sont plus fixés une fois pour toutes dans le SGBD mais ils sont fabriqués dynamiquement selon les besoins, donc pas de "navigation" dans le labyrinthe des pointeurs.

---Des langages de haut niveau pour décrire et manipuler les données. Cela permet l'utilisation des opérateurs logiques de la théorie des ensembles dans le langage d'interrogation pour que le système soit capable de comprendre correctement une question.

---Des logiciels peuvent fonctionner dans des environnements différents. Cela signifie que, sur les très gros ordinateurs, on trouvera toutes les fonctionnalités décrites antérieurement, alors que, sur des micro-ordinateurs, on trouvera des versions simplifiées.

---La standardisation du langage conceptuel.

L'ORGANISATION RELATIONNELLE

Le modèle relationnel des bases de données est beaucoup plus proche du raisonnement humain, que les modèles hiérarchique ou réseau.

Par exemple, lorsque l'on nous pose une question, il va de soi que nous n'allons pas consulter la totalité de notre mémoire, ni même toutes les informations que nous possédons sur un sujet donné, avant de donner notre réponse. Pour l'homme, c'est le contenu de la question qui va servir pour déduire la réponse. Il y a un processus déductif qui est à l'opposé du processus inductif utilisé dans la recherche sur une base de données hiérarchisée.

Lorsqu'on nous demande quelle est notre profession, il est évident que nous n'avons pas besoin de passer mentalement en revue tous les métiers que nous connaissons pour trouver le nôtre. Nous le trouvons immédiatement parce que la question est posée de telle façon que les mots "profession" et "notre" réduisent le champ de recherche à son strict minimum.

Les bases de données relationnelles fonctionnent donc selon ce principe: c'est le contenu de la question qui va déterminer quels sont les chemins d'accès à utiliser et les liens à établir.

Ces liens, ou pointeurs, ne seront plus fixés une fois pour toutes dans le SGBD mais ils seront fabriqués dynamiquement selon les besoins. Mais cela n'est pas suffisant car il faudra que le langage d'interrogation du système soit capable de comprendre correctement une question. Pour cela, une seule possibilité: l'utilisation des opérateurs logiques de la théorie des ensembles.

VOCABULAIRE

RELATION : C'est la liaison entre deux ou plusieurs ensembles d'informations caractérisé par un nom.

Exemple de représentation :

FLORE (CODESPECE, NOMGENRE, NOMESPECE)

CAMPAGNE (NUMCAMPAGNE, RELEVE, CODESPECE, NOTESPECE)

R (A , B , C , D)

Les liaisons qui existent entre les informations, sont finalement plus importantes que les informations elles mêmes. Donc dans le modèle relationnel l'important sont les liaisons entre les informations, en plus, toutes les relations sont à priori des relations quelconques et en général de type n-m.

OPERATIONS SUR LES RELATIONS : Les opérations sur les relations donnent comme résultat des relations, n'importe quel résultat peut être considéré comme une relation.

ATTRIBUT : Les attributs d'une relation sont les ensembles des informations qui participent à cette relation (colonnes d'une relation caractérisées par un nom).

Par exemple:

REALISATION (NUMCAMPAGNE, NOMCAMPAGNE, AUTEUR, DATE)

R (A , B , C)

"A" est un attribut, "B" et "C" aussi, donc la relation R comporte trois attributs. La relation REALISATION comporte quatre.

DOMAINE : C'est l'ensemble des valeurs possibles que peut prendre un attribut.

Dans une opération, à l'intérieur d'une même relation on peut avoir deux attributs prenant ses valeurs dans un même domaine.

Par exemple :

TAXON (GENRE, ESPECE, SOUS-ESPECE, VARIETE)

Dans ce cas, on admet qu'il y a quatre chaînes de caractères par ordre alphabétique, donc les domaines des attributs sont les mêmes (il y a même des noms de sous-espèces qui ont des noms d'espèces, etc).

Dans le modèle relationnel on distingue bien ces deux mots (attribut et domaine) et on ne peut pas avoir deux attributs qui portent le même nom.

PREMIERE REGLE: L'ordre des attributs est non significatif, c'est-à-dire, que les attributs peuvent être dans n'importe quel ordre, parce qu'il n'y a pas de particularité à priori d'un attribut par rapport à un autre.

DEUXIEME REGLE: Il faut nécessairement que les attributs aient des noms différents dans une même relation. Tous les attributs doivent être distincts.

DEGRE : Le degré d'une relation est le nombre d'attributs de la relation.

Par exemple:

R1 (A , B , C , D) ⟵ c'est une relation de degré quatre

R2 (A , B) ⟵ c'est une relation binaire (degré deux)

QUANTIFICATION (NUMCAMPAGNE, CODESPECE, NOMBRELEVES) ⟵ degrés trois

N-UPLETS : Sont les éléments d'une relation de degré "n". Pour une relation R (A,B,C), ils sont notés (a,b,c).

DUPLET ----- R (a , b)

TRIPLET ----- R (a , b , c) ..... ETC.

Par exemple, dans la relation:

FLORE (CODESPECE, NOMGENRE, NOMESPECE) .....un triplet sera

(5, Abies, alba)

CONTENU D'UNE RELATION : Le contenu d'une relation est le sous-ensemble du produit cartésien des domaines des attributs de la relation, pour lequel la relation a une signification (signification décidée par l'utilisateur).

Le produit cartésien des domaines des attributs est tout simplement toutes les combinaisons possibles de toutes les valeurs de chaque domaine.

Par exemple:

R (A , B , C)

Si on a 10 valeurs du premier domaine, 10 valeurs du deuxième domaine et 5 valeurs du troixième; le produit cartésien sera:

10 * 10 * 5 = 500

..... Mais toutes ces combinaisons ne correspondent pas à la réalité, et c'est seulement à l'utilisateur de savoir si telle ou telle combinaison est vraie ou fausse.

Si pour une combinaison la relation est vraie, alors cette combinaison appartient au contenu de la relation. Par contre, si pour une combinaison la relation est fausse, hélas, cette combinaison n'appartient pas au contenu de la relation.

Exemple:

TAXON	CODE	NOM	CATECOLOG
	2211	Lumbricus terrestris	anécique
	3121	Dendrobaena rubida	épigé
	3611	Octolasiom cyaneum	endogé
	4311	Eisenia fetida	épigé
	6112	Hormogaster redii	anécique

Dans cette relation on suppose qu'il y a cinq triplets; l'attribut CODE possède 5 valeurs différentes, l'attribut NOM aussi et l'attribut CATECOLOG 3.

Le produit cartésien pour cette relation sera:

5 * 5 * 3 = 75 .....mais ce sont les cinq triplets précédents qui ont seulement une signification dans la réalité. Le triplet (6112, Octolasiom cyaneum, épigé) par exemple, n'appartient pas au contenu de la relation (n'a aucune signification).

CARDINALITE : C'est la taille de la relation, le nombre de n-uplets présents dans la relation. Le temps de traitement dépend de la cardinalité des relations, il faut chercher un compromis avec le degré au niveau de la conception des relations.

Dans l'exemple précédent la cardinalité de la relation est de cinq.

CLE : La clé correspond à un ou plusieurs attributs de la relation où la validité de l'unicité des n-uplets sera verifiée. La connaissance de la clé permet d'identifier un n-uplet unique de la relation considérée.

La règle d'unicité des n-uplets joue normalement sur l'ensemble des valeurs des attributs, c'est-à-dire, sur la combinaison de toutes les valeurs, mais l'unicité des n-uplets sera verifiée seulement sur la CLE.

Par exemple, dans la relation:

TAXON (CODE , GENRE, ESPECE, SOUS-ESPECE, VARIETE, CATECOLOG)

la clé est CODE et le n-uplet .....

(2661, LUMBRICUS, RUBELLUS, FRIENDOIDES, TYPICA, ANECIQUE)

..... exprime le fait qu'il existe un taxon qui a un certain code qui correspond à un certain nom et qui appartient à une certaine catégorie écologique. Donc qu'il y a une combinaison unique de ces six valeurs qui indique que la relation est vraie, parce qu'elle correspond à un seul taxon.

Autre exemple:

INDIVIDU (N° PRELEVEMENT , N° INDIVIDU , N° TAXON, STADE, ETAT, POIDS)

Dans cette relation la clé ne peut pas être seulement le N° de l'individu, pour la raison que le même numéro peut être utilisé dans plusieurs prélèvements; elle ne peut pas être non plus le N° du prélèvement tout seul, parce qu'un même numéro de prélèvement peut être associé à différents individus. Par contre l'ensemble des deux va former la clé et la relation sera vraie si elle correspond à un seul individu.

Dans une relation il ne peut pas y avoir deux n-uplets identiques. Donc il ne peut y avoir qu'un seul n-uplet correspondant à une combinaison donnée des valeurs des attributs. Si on essaie de rentrer deux n-uplets avec la même clé, même si la suite est différente, le système refoulera l'opération.

Il faut bien comprendre que la clé d'une relation n'est pas une clé d'accès, mais une clé de vérification de l'unicité des n-uplets; l'accès aux n-uplets d'une relation peut se réaliser par n'importe lequel des attributs de la relation.

FORMES NORMALES

Les formes normales sont des contraintes sur lesquelles doit se replier la relation pour permettre une meilleure possibilité de traitement du contenu des relations.

PREMIERE FORME NORMALE:

Une relation sera dite en première forme normale si tous les attributs de cette relation sont en valeur unique et non multiple, donc si le contenu de la relation peut être représenté par un tableau bidimensionnel, ayant autant de colonnes que d'attributs et autant de lignes que de n-uplets dans la relation.

Les systèmes basés sur le modèle relationnel ne traitent que des relations qui sont toujours dans la première forme normale.

NORMALISATION D'UNE RELATION:

Dans la realité on se trouve parfois avec des relations qui ne sont pas dans la première forme normale. Donc, il va falloir faire ce qu'on appelle la normalisation de ces relations, de façon à les transformer en première forme normale.

La normalisation va consister en fait, à décomposer la relation initiale en deux relations qui sont de degré inférieur à la relation de départ.

Par exemple:

INDIVIDU (INSEE , NOM, PRENOM, CONJOINT, ENFANT)

Dans cet exemple, la clé est le numéro d'INSEE parce que par individu elle est unique. Cette relation présente une anomalie qui consiste au fait que le dernier attribut (ENFANT) est un attribut à valeur multiple. Un attribut à valeur multiple dans le sens qu'on va pouvoir associer à chaque individu de 0 à N valeurs.

Alors la normalisation de cette relation s'effectue de la manière suivante:

INDIVIDU (INSEE , NOM, PRENOM, CONJOINT)

DESCEND (INSEE , ENFANT)

Dans ce cas, il faudra deux accès pour récupérer les informations sur un individu; on sera obligé d'accéder à la première relation pour pouvoir récupérer son NOM, PRENOM, et les informations sur son conjoint, en suite accéder à la deuxième relation pour pouvoir récupérer les information de ses enfants.

DEUXIEME FORME NORMALE:

Une relation est en deuxième forme normale, si elle est déjà en première forme normale (condition nécessaire mais pas suffisante). Il faut en plus que l'ensemble des attributs ne formant pas la clé, dépendent globalement des attributs qui forment la clé et non partiellement.

On voit donc apparaître ici une nouvelle notion: la notion de dépendance .

Par exemple:

APPROVISIONNEMENT (ARTICLE , FOURNISSEUR , PRIX, ADRESSE)

Dans cet exemple, la clé ne peut pas être seulement l'article, pour la raison que le même article peut être produit par plusieurs fournisseurs; elle ne peut pas être non plus le fournisseur tout seul, parce qu'un même fournisseur peut être associé à différents articles; par contre l'ensemble de deux va former la clé.

Dans cette relation l'attribut ADRESSE a une dépendance partielle vis-à-vis de la clé. L'adresse dépend seulement du fournisseur et non de l'article. Alors si le fournisseur change d'adresse,il faudra la modifier dans tous les n-uplets de la relation.

La normalisation de cette relation se fait en la coupant en deux.

APROVISIONNEMENT (ARTICLE , FOURNISSEUR , PRIX)

FOURNISSEUR (FOURNISSEUR , ADRESSE)

TROISIEME FORME NORMALE:

Une relation est en troisième forme normale, si elle est déjà en deuxième forme normale, et si la dépendance des attributs ne formant pas la clé, vis-à-vis de la clé est une dépendance directe et non transitive.

Exemple d'une base de données du personnel d'une entreprise d'informatique.

INDIVIDU (MATRICULE , NOM, PROJET, DATFIN-PROJET)

La clé de cette relation est matricule, parce qu'il va y avoir seulement un seul n-uplet par individu et que le matricule par individu est unique.

Cette relation est en première forme normale parce qu'aucun attribut n'est repétitif; cette relation est aussi en deuxième forme normale pour la raison que si la clé est formée d'un seul attribut, la dépendance des autres attributs ne formant pas clé vis-à-vis de la clé ne peut être que global et non partiel.

Dans l'exemple, un matricule donné est associé à un seul individu, qui a donc un seul nom et qui est affecté à un seul projet, donc là, la dépendance est bien une dépendance directe, c'est-à-dire que le matricule fixe automatiquement à un moment donné, le projet.

Date-fin-projet, correspond à la date à laquelle l'individu va être disponible; disponible parce que le projet sera terminé. Mais, date-fin-projet dépend du projet, alors c'est une dépendance non directe de la clé, là encore il y a redondance d'information, parce que si plus d'un individu est associé à ce projet, cette date sera répétée dans autant de n-uplets que d'individus seront associés à ce projet, si on modifie la date-fin-projet, alors il faudra la modifier dans tous les individus qui vont être liberés à ce moment là, et on se retrouvera avec les mêmes inconvénients que si la relation n'était pas en deuxième forme normale.

Comment normaliser cette relation?

INDIVIDU (MATRICULE , NOM, PROJET)

PROJET (PROJET , DATE-FIN)

Pour gérer le personnel, il faut savoir à quelle date tel individu sera disponible et il faudra accèder à deux relations. A la première pour savoir à quel projet il a été affecté, à la deuxième pour savoir à quelle date le projet sera terminé, c'est-à-dire, à quelle date l'individu sera disponible.

LES OPERATIONS SUR LES RELATIONS

Les principaux opérateurs relationnels sont l'union, l'intersection, la différence, le produit cartésien, la sélection, la projection et la composition.

L'UNION

C'est une opération qui s'effectue entre deux relations

---Il faut que les deux relations soient de même degré.

---Le degré de la relation résultat est le même que le degré des deux relations de départ.

---La cardinalité de la relation résultat est inférieure ou égale à la somme des deux cardinalités de départ. Tous les n-uplets de la deuxième relation qui sont les mêmes que dans la première, sont exclus (il faut respecter la règle d'unicité des n-uplets).

---Le domaine d'un attribut résultat va être l'union des deux domaines des attributs de départ.

---Il n'y a aucune obligation que les attributs soient respectivement les mêmes. Si les attributs ne portent pas les mêmes noms dans les deux relations de départ, il faudra déterminer quels sont les noms des attributs dans la relation résultat.

Exemple:

relations de départ	UNION	relation résultat
R (A, B, C)		
	V = R U S	V (G, H, I)
S (D, E, F)		

Explication:

première relation

	R	A	B	C
*		a1	b1	c1
		a2	b2	c2
		a3	b3	c3
*		a4	b4	c4
		a5	b5	c5

deuxième relation

	S	D	E	F
*		a1	b1	c1
*		a4	b4	c4
		a6	b6	c6
		a7	b7	c7

relation résultat

V	G	H	I
	a1	b1	c1
	a2	b2	c2
	a3	b3	c3
	a4	b4	c4
	a5	b5	c5
	a6	b6	c6
	a7	b7	c7

L'INTERSECTION

C'est une opération qui s'effectue entre deux relations.

---Dans ce cas on ne récupère que les n-uplets qui sont à la fois dans la première et dans la deuxième relation. On trouvera dans la relation résultat seulement les n-uplets de la première relation qui sont identiques dans la deuxième, parce qu'il y a la règle d'unicité qui joue.

---Il faut que les deux relations de départ soient du même degré.

---Le degré de la relation résultat est le même que le degré des deux relations de départ.

---Le domaine des attributs de la relation résultat est dans la même position que les domaines des attributs de départ. Si les domaines des deux relations de départ sont disjoints la relation résultat est vide, dans ce cas, est inutile de faire l'opération, même chose si tous les n-uplets sont identiques.

---La cardinalité de la relation résultat est égale ou inférieure à la cardinalité de la plus petite relation de départ.

Exemple:

relations de départ	INTERSECTION	relation résultat
R (A, B, C)		
	V = R ∩ S	V (G, H, I)
S (D, E, F)		

Explication:

première relation

	R	A	B	C
*		a1	b1	c1
		a2	b2	c2
		a3	b3	c3
*		a4	b4	c4
		a5	b5	c5

deuxième relation

	S	D	E	F
*		a1	b1	c1
*		a4	b4	c4
		a6	b6	c6
		a7	b7	c7

relation résultat

V	G	H	I
	a1	b1	c1
	a4	b4	c4

LA DIFFERENCE

C'est une opération qui s'effectue entre deux relations. La différence est un opérateur non commutatif; l'ordre des relations opérandes est donc important.

---Dans ce cas on ne récupère que les n-uplets de la première relation qui n'apparaissent pas dans la deuxième. On trouvera dans la relation résultat seulement les n-uplets de la première relation qui ne se répètent pas dans la deuxième.

---Il faut que les deux relations de départ soient du même degré.

---Le degré de la relation résultat est le même que le degré des deux relations de départ.

---La cardinalité de la relation résultat est inférieure ou égale à la cardinalité de la première relation de départ.

---Le domaine des attributs de la relation résultat est dans la même position que les domaines des attributs de la première relation de départ.

Exemple:

relations de départ	DIFFERENCE	relation résultat
R (A, B, C)		
	V = R - S	V (G, H, I)
S (D, E, F)		

Explication:

première relation

	R	A	B	C
*		a1	b1	c1
		a2	b2	c2
		a3	b3	c3
*		a4	b4	c4
		a5	b5	c5

deuxième relation

	S	D	E	F
*		a1	b1	c1
*		a4	b4	c4
		a6	b6	c6
		a7	b7	c7

relation résultat

V	G	H	I
	a2	b2	c2
	a3	b3	c3
	a5	b5	c5

LE PRODUIT CARTESIEN

C'est une opération qui s'effectue entre deux relations.

---Pour construire la relation résultat, on prend chaque n-uplet de la première relation de départ auquel on "concatène" chaque n-uplet de la deuxième relation.

---Il n'est pas nécessaire que les deux relations de départ aient le même degré.

---Le degré de la relation résultat sera égal à la somme des degrés des relations de départ.

---La cardinalité de la relation résultat est égale au produit des cardinalités des relations de départ.

---Le domaine des attributs de la relation résultat est dans la même position que les domaines des attributs des deux relations de départ.

Exemple:

relations de départ	PRODUIT CARTESIEN	relation résultat
R (A, B, C)		
	V = R * S	V (A, B, C, D, E)
S (D, E)		

Explication:

première relation

R	A	B	C
	a1	b1	c1
	a2	b2	c2
	a5	b5	c5

deuxième relation

S	D	E
	d1	e1
	d2	e2

relation résultat

V	A	B	C	D	E
	a1	b1	c1	d1	e1
	a1	b1	c1	d2	e2
	a2	b2	c2	d1	e1
	a2	b2	c2	d2	e2
	a5	b5	c5	d1	e1
	a5	b5	c5	d2	e2

LA SELECTION

C'est une opération qui s'effectue à l'intérieur d'une même relation et qui correspond à l'élimination de n-uplets (suppression de lignes).

---A partir d'une relation on peut en créer une autre, tel qu'une ou plusieurs conditions soient satisfaites. On peut utiliser les opérateurs de comparaison et logiques, même en les combinant entre eux.

---La relation résultat est créée avec les n-uplets de la relation de départ qui obéissent à la ou les conditions imposées.

---La relation résultat a la même allure que la relation de départ, c'est-à-dire, qu'elle a les mêmes degré et domaines; à priori elle a aussi les mêmes attributs.

---La cardinalité de la relation resultat va être comprise entre 0 (si aucun n-uplet satisfait la ou les conditions imposées) et la cardinalité de la relation de départ (si tous les n-uplets satisfont la ou les conditions). En règle générale elle sera intermédiaire entre les deux.

Exemple:

relation de départ ----- R (A, B, C)

SELECTION ----- S égal sous-ensemble de R, telle qu'une condition

relation résultat ----- S (A, B, C)

Explication:

relation de départ

R	A	B	C
	a1	b1	c2
	a2	b2	c4
	a3	b1	c5
	a4	b3	c6
	a5	b4	c7
	a6	b1	c8
	a7	b1	c9

S = R / A > a2 ET B = b1

relation résultat

S	A	B	C
	a3	b1	c5
	a6	b1	c8
	a7	b1	c9

LA PROJECTION

C'est une opération qui s'effectue à l'intérieur d'une même relation et qui correspond à l'élimination d'attributs (suppression de colonnes).

---En considérant un à un les n-uplets d'une relation de départ on construit une relation résultat en ne gardant que les attributs appartenant à la projection. Donc, les attributs projetés sont les mêmes que les attributs de départ.

---La relation résultat a un degré inférieur au degré de la relation de départ.

---La cardinalité de la relation résultat peut être égale à la cardinalité de la relation de départ si les attributs qui forment la clé sont conservés dans la projection, sinon elle peut être inférieure (règle d'unicité des n-uplets).

---Les domaines des attributs de la relation résultat sont les mêmes que ceux de la relation de départ.

Exemple:

relation de départ ----- R (A, B, C, D)

PROJECTION ----- S = PROJECTION (A , C) R

relation résultat ----- S (A , C)

Explication:

relation de départ

R	A	B	C	D	E
	a1	b1	c1	d1	e1
	a2	b2	c2	d2	e2
	a3	b3	c3	d3	e3
	a6	b6	c6	d6	e6

relation résultat

S	A	C
	a1	c1
	a2	c2
	a3	c3
	a6	c6

LA COMPOSITION

C'est une opération qui s'effectue entre deux relations.

---On privilégie un attribut dans chaque relation de départ (on les appelle pivots).

---On cherche l'égalité des valeurs entre l'attribut pivot de la première relation et l'attribut pivot de la deuxième relation de départ.

---Si l'égalité des valeurs existe entre les attributs pivots des deux relations, on fabrique dans la relation résultat autant de n-uplets que ceux qu'on a trouvé.

---Le degré de la relation résultat est égal à la somme des degrés des deux relation de départ - 1 (composition naturelle).

---La cardinalité varie entre les extrêmes 0 (zéro) et le produit des deux cardinalités. Dans la pratique on va trouver un résultat intermédiaire entre les deux.

---Les domaines des attributs de la relation résultat sont les mêmes que ceux des relations de départ.

Exemple:

relations de départ — COMPOSITION — relation résultat

R (A, B, C)

S (D, E)

V = R (B) * S (D)

V (A, B, C, E)

Explication:

première relation

R	A	B	C
	a1	b1	c1
	a1	b2	c1
	a2	b1	c2
	a3	b3	c1

deuxième relation

S	D	E
	d1	e1
	b1	e2
	d2	e1
	b3	e2
	b3	e3

relation résultat

V	A	B	C	D	E
	a1	b1	c1	b1	e2
	a2	b1	c2	b1	e2
	a3	b3	c1	b3	e2
	a3	b3	c1	b3	e3

B = D alors ------>

V	A	B	C	E
	a1	b1	c1	e2
	a2	b1	c2	e2
	a3	b3	c1	e2
	a3	b3	c1	e3

STOCKAGE ET ACCES AUX DONNEES DANS UN SYSTEME RELATIONNEL

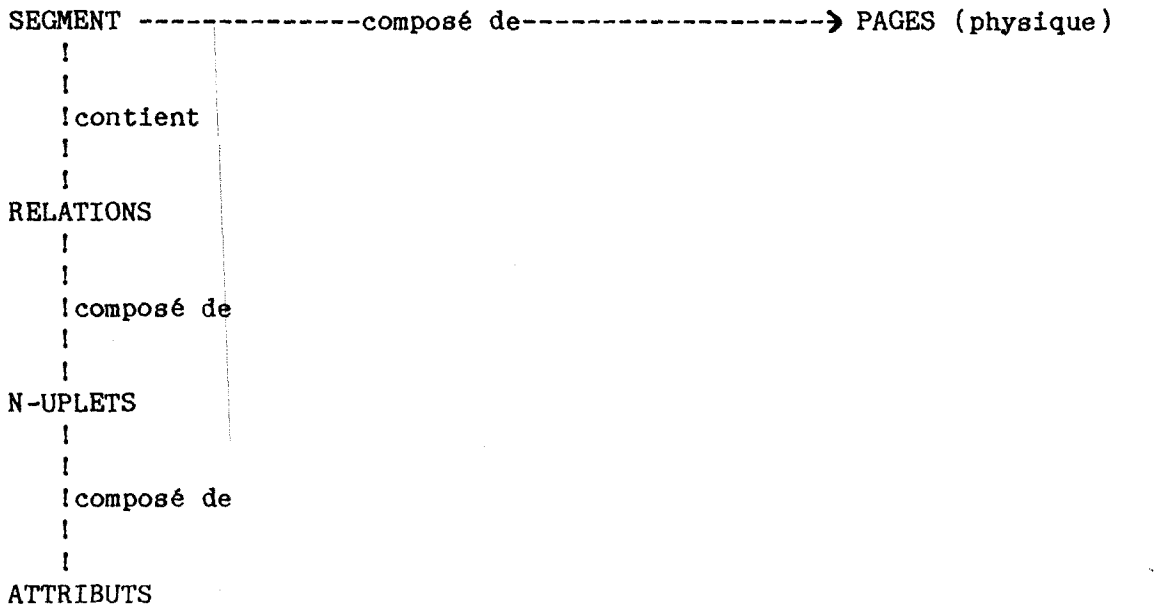
Même si au niveau relationnel les données apparaissent pour l'usager comme des tableaux de valeurs, il va sans dire que le stockage de ces données n'est pas, en général, réalisé sous forme de simples tableaux. On imagine mal en effet une base de données relationnelle composée d'un fichier par relation, fichier sur lequel ne serait possible qu'un accès séquentiel. Les performances d'un tel système seraient désastreuses et dénatureraient les buts du modèle relationnel.

Dans le cadre des SGBD relationnels, l'aspect performance est un facteur déterminant. Les prototypes les plus récents ont cependant prouvé qu'il était possible d'obtenir, avec la technologie actuelle, des performances raisonnables. De plus, il est incontestable que le modèle relationnel apporte une simplification telle dans la description et l'utilisation de la base que parfois une légère dégradation des performances est tolérable.

Le stockage d'une base de données relationnele est en général fait dans un espace de mémoire secondaire (disque) subdivisé logiquement en différentes zones et s'appuyant sur des méthodes d'accès qui sont soit propres au SGBD lui-même soit fournies par les système d'exploitation sous lequel fonctionne le SGBD.

Ainsi, l'espace de stockage se présente en général comme une ensemble de plusieurs segments. Un segment est un espace virtuel de stockage qui se compose physiquement de plusieurs pages de taille fixe; tous les segments n'ont pas la même taille (nombre de pages maximum) qui est une paramètre choisit à la création de la base. La page est l'unité d'Entrée/Sortie, c'est-à-dire que les transferts entre mémoire centrale et mémoire secondaire (disque) ne se font que par bloc physique de taille fixe, variant entre 512 à 4096 octets. Chaque page physique a un numéro qui permet de la repérer.

Une relation est stocké dans un seul segment et peut donc s'étendre sur plusieurs pages du même segment. Une relation est un ensemble de n-uplets, chacun d'eux étant composé de plusieurs attributs.



Pour qu'une telle base soit exploitable, il faut, en particulier, disposer de chemins d'accès: par exemple un index qui est lui-même une relation contenant des "n-uplets" qui "pointent" vers celles d'une autre relation. En général, il y a deux type de chemins d'accès: les index sur une relation et les liens entre deux relations.

Le sous-système de stockage et d'accès permet grâce au concept de balayage la recherche de n-uplets dont on connaît tout ou partie des valeurs des attributs.

Les opérations qu'on peut effectuer sur une relation sont: l'insertion, la suppression, la modification et la recherche d'un n-uplet. On suppose que chaque n-uplet peut être repéré par un numéro qui peut être soit absolu, soit composé d'un numéro de page et d'un déplacement dans la page.

CONCLUSION

Pour concevoir de façon naturelle le schéma conceptuel d'une base, il faut une approche méthodique, par exemple:

a) création d'un dictionnaire de données;

b) définition des différents types de données, une seule fois pour chacune, et les liens logiques existant entre elles;

c) création d'une façon naturelle des relations correspondantes;

d) prendre les relations une par une et déterminer les attributs qui formeront la clé;

e) mettre en première forme normale les relations qui ne le sont pas (décomposition en deux relations), puis en deuxième et troisième forme normale si nécessaire.

BIBLIOGRAPHIE :

--- Balmes, R. 1984-Les bases de données et leurs différents modèles. M.S. N°s. 41, 42 et 43.

--- Bouché, M. 1972-Lombriciens de France écologie et systématique. INRA.

--- Delobel, C. et Adeba, M. 1982-Bases de données et systèmes relationnels. Dunod.

--- Filliatre, B. 1983/84-Notes du cours Bases de Données. CNAM.

--- Rein, R., 1971-L'analyse en informatique de gestion. Dunod.

NOTE :

Des exemples concrets, sur l'organisation générale des données écologiques seront posés dans la deuxième partie de ce document appelé:

"SCHEMA CONCEPTUEL DE LA BASE DE DONNEES E C O R D R E".