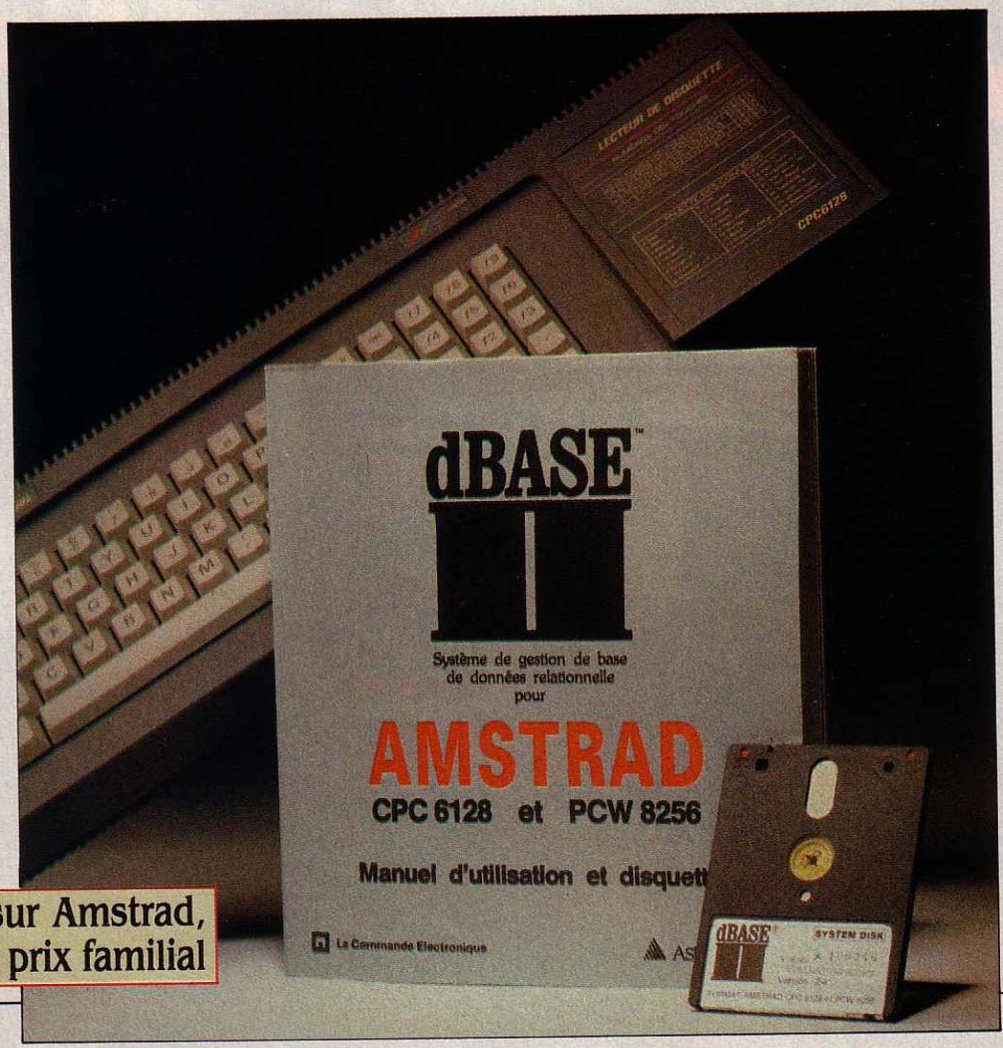
COMPRENDRE

Visite guidée de dBase II sur Amstrad, un outil professionnel à un prix familial

Jusqu'à présent, les logiciels de gestion de bases de données professionnels coûtaient plusieurs milliers de francs. Aujourd'hui, pour la première fois, l'un de ces logiciels, dBase II d'Ashton Tate, est disponible pour seulement 790 F TTC (manuel de 500 pages compris) sur un ordinateur familial : l'Amstrad CPC 6128 (il marche aussi sur le PCW 8256, l'ordinateur spécialisé de traitement de texte d'Amstrad). Vu que dBase II ne se vend pratiquement plus pour des ordinateurs professionnels (il est destiné aux anciennes machines 8 bits, et a été remplacé par une version 16 bits, dBase III), Ashton Tate n'avait rien à perdre en l'adaptant, à un prix cassé, sur un petit micro très populaire : les frais de mise au point de ce logiciel sont amortis depuis longtemps. Du coup, l'Amstrad acquiert une coloration professionnelle, et le couple qu'il forme avec dBase II est un outil suffisamment puissant et complet pour servir de support à la visite guidée dans le monde des bases de données que nous vous proposons ici.

DBASE II EST SANS AUCUN DOUTE le logiciel de gestion de base de données qui fut le plus populaire ces dernières années. Apparu en 1980, il a été le premier à offrir sur des petits ordinateurs la puissance et la facilité d'emploi que l'on pouvait trouver alors sur des systèmes beaucoup plus importants ; dBase II a été adapté à la fin des années 1970 à partir d'un logiciel tournant sur un Univac-1108 sous le nom de JPLDIS. Il a depuis été adapté sur le plus grand nombre de micro-ordinateurs équipés de microprocesseurs 8 ou 16 bits et l'on estime à plus de 500 000 le nombre d'exemplaires vendus à travers le monde à ce jour. Ce n'est qu'à partir de 1983 qu'ont commencé à apparaître des produits vraiment capables de rivaliser avec lui. Pour faire face à cette concurrence, Ashton Tate a développé une version plus puissante de dBase II apparue en 1984 sous le nom de dBase III, mais ne pouvant fonctionner que sur des ordinateurs possédant un microprocesseur 16 bits tel que l'IBM PC. Malgré ses six ans d'âge, dBase II n'a pris que très peu de rides et l'on peut lui prédire un large succès sur un micro-ordinateur comme Amstrad.

Il y a beaucoup de controverses sur la définition *stricto sensu* du terme « base de données ». La définition la plus généralement ac-



LES BASES DE DONNÉES

En l'absence d'une définition clairement établie, et pour simplifier les choses au lecteur profane, nous considérerons dans cet article d'initiation que dBase II est bien un gestionnaire de base de données.

ceptée les désigne comme un ensemble d'informations organisées de façon à être facilement retrouvées, triées, etc.

Pratiquement une base de données est constituée de plusieurs fichiers organisés de manière à supprimer les redondances d'information. Chaque fichier contient des enregistrements et chaque enregistrement contient des rubriques. Par exemple, une base de données destinée à la gestion d'un commerce contiendra un fichier des clients et un fichier de factures ; les enregistrements du fichier « Clients » contiendront les rubriques nom, prénom, adresse, etc., les enregistrements du fichier « Facture » contiendront les rubriques

Photos Armand BORLANT

La création d'un fichier dBase II. Il faut indiquer le nom des rubriques, leur type et la longueur qu'ils occupent.

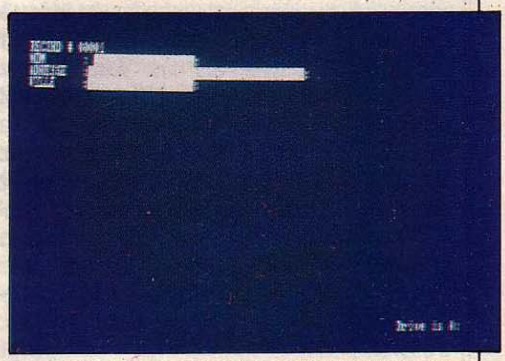
article commandé, quantité, date, etc. Le nom du client ne sera pas répété dans le fichier de facture. A chaque client, on associera dans le fichier des factures un certain nombre d'enregistrements. Cette association se matérialise par ce que les informaticiens appellent un lien. Pour les puristes, une base de données n'est même pas définie par un ensemble de fichiers, mais par un ensemble d'informations entre lesquelles existent des relations. Ces puristes feront la différence entre un gestionnaire de fichier comme dBase II et un vrai système de base de données dit relationnel.

Pour construire votre premier fichier, il faut donc d'abord en définir la structure. Il vous suffit de taper la commande CREATE pour effectuer cette opération. dBase II vous demande alors de donner un nom au fichier que vous allez créer (exactement comme si vous aviez un classeur sur lequel vous auriez marqué « clients » ou bien « fournisseurs »), puis de nommer toutes les rubriques que vous désirez voir figurer dans ce fichier. Une rubrique – ou champ – correspond aux mentions que vous pourriez reporter sur une fiche cartonnée : nom, prénom, adresse, etc.

Ensuite, il faut indiquer à dBase II si telle ou telle rubrique sera destinée à contenir du texte, des chiffres ou bien une valeur logique (vrai ou faux). Enfin la dernière information requise est l'espace (en nombre de caractères) que vous désirez réserver pour chaque rubrique.

Création de rubriques

C'est important, car cela conditionne l'espace pris par le fichier sur la disquette servant de support. Pour reprendre l'exemple du carnet d'adresses, vous pouvez choisir comme noms de rubrique NOM, ADRESSE et VILLE, les déclarer de type texte, puis réserver 20 caractères pour la rubrique NOM, 40 pour la rubrique ADRESSE et 20 pour la rubrique VILLE. Notez qu'une rubrique qui contient les mots « 19 rue Dufour » est à déclarer de type texte. Pour l'ordinateur en effet, dans ce cas précis, les chiffres 1 et 9 sont des caractères comme les autres, des lettres d'une forme particulière. En revanche, dans un fichier de comptabilité, une rubrique qui contiendrait « 20,50 F » serait à déclarer de type numérique, puisque l'ordinateur a besoin de faire des calculs dessus. C'est pourquoi on parle de caractères alphanumériques pour décrire l'ensemble de l'alphabet, des signes de ponctuation divers et des chiffres « inertes ». Par ailleurs, même si vous ne remplissez pas à chaque fois l'espace que vous avez fixé pour chaque rubrique, votre fichier occupera sur la disquette une place correspondant à la longueur maximum. Ces manipulations sont indubitablement une contrainte qui n'existe pas dans le cas d'un fichier classique sur papier, mais c'est le prix à payer pour pouvoir exploiter les données de la base ultérieurement. Une fois la structure créée (photo ci-contre), dBase II propose de commencer la saisie des données immédiatement. Si vous répondez par l'affirmative, le nom de toutes les rubriques créées apparaît, vous invitant à taper les premières données (photo ci-dessous). Si vous répondez par la négative, vous quittez momentanément le fichier que vous venez de créer et qui ne contient encore aucune donnée. Seule la structure est conservée ; vous pourrez introduire des données ultérieurement avec la commande APPEND. Dans notre exemple, le fichier ainsi constitué ressemble au schéma 1. Il s'agit d'une organisation très simple d'un tableau à deux dimensions où les enregistrements prennent place les uns après les autres, séquentiellement, au fur et à mesure qu'on les tape au clavier. Chaque colonne constitue ce que l'on appelle un ensemble de champs (c'est-à-dire de rubriques) et



La saisie des données dans le fichier. L'espace disponible est matérialisé par une zone blanche.

chaque rangée forme un enregistrement (c'est l'équivalent d'une fiche). Un fichier dBase II est toujours associé à un en-tête qui contient un certain nombre d'informations indispensables à l'exploitation (voir schéma 2). Ces informations sont à tout moment accessibles : il suffit de taper la commande DISPLAY STRUCTURE. On obtient alors une vue générale du fichier, la date de la dernière mise à

jour, le nombre de champs et leur type, ainsi que le nombre total d'enregistrements déjà réalisés.

Après la saisie des données, vient le moment de l'exploitation. Les usages courants d'un tel logiciel concernent plus particulièrement la recherche d'un enregistrement précis, le tri, c'est-à-dire le classement par ordre croissant ou décroissant (par ordre alphabétique, par exemple), et l'impression sur papier de tout ou partie du fichier. Lorsque l'on recherche un enregistrement particulier dans une base de données, la solution a priori la plus simple consiste à lire tout le fichier du début jusqu'à la fin en vérifiant à chaque fois si l'enregistrement correspond à celui que l'on recherche. Si cette solution peut paraître satisfaisante pour les fichiers de petite taille, c'est en revanche peu imaginable pour ceux atteignant une dimension importante. Cela prendrait en effet beaucoup trop de temps. Pour remédier à cela, dBase II possède une fonction d'indexation qui obéit au même principe qu'une table des matières. Pour trouver le présent article dans SVM, vous avez deux possibilités : soit lire toutes les pages précédentes à partir du début du journal, soit vous reporter au sommaire qui mentionne le numéro de la page recherchée. La première méthode s'appelle la recherche séquentielle ; la seconde, la recherche indexée. dBase II fonctionne exactement de la même façon. La commande INDEX permet de créer l'équivalent du sommaire, qui contiendra d'un côté le nom des rubriques sur lesquelles vous avez fait l'indexation (par exemple le titre des articles par ordre alphabétique) et le numéro des enregistrements correspondant (comparable aux numéros de page). dBase II vous donne la possibilité de créer ainsi jusqu'à sept index à partir de différentes rubriques et de les utiliser indifféremment selon les cas. Vous pouvez aussi créer des index portant simultanément sur plusieurs rubriques ; par exemple, la commande INDEX ON NOM+VILLE signifie que les données vont être indexées par ordre alphabétique des villes, puis par ordre alphabétique des noms au sein de chaque ville. Mais la création d'un index n'est pas tout. Encore faut-il organiser les données à l'intérieur de l'index.

Recherche dichotomique

L'index est lui-même un fichier dont les enregistrements sont simplement constitués de la rubrique indexée et du numéro (adresse) de l'enregistrement correspondant dans le fichier des données. La vitesse de recherche dépend de la capacité du système de gestion de données à parcourir cet index plus ou moins rapidement. Quel que soit le système de base de données, l'efficacité de la méthode employée se fait au détriment de la place mémoire et de la puissance nécessaires à son fonctionnement. Les différents procédés sont toujours des compromis. A partir de 1970, des chercheurs américains ont développé un certain nombre d'algorithmes permettant d'organiser et de rechercher des données. La première méthode est ce que l'on appelle la recherche dichotomique. Son principe est très simple, il réclame que les données soient organisées suivant un certain ordre (alphabétique en règle générale). C'est l'organisation typique d'un dictionnaire. Supposons que l'on recherche le mot « écrire » dans un dictionnaire de 688 pages. La solution la plus rapide consiste à ouvrir le dictionnaire exactement au milieu (page 344), où l'on tombe sur le mot « invincible ». Comme l'on sait que les mots sont rangés par ordre alphabétique, il est très facile de déterminer que le mot « écrire » se trouve dans la première moitié du dictionnaire (entre « a » et « invincible ». Il suffit alors de répéter l'opération précédente, c'est-à-dire d'ouvrir le dictionnaire au milieu de la première moitié (page 172) où l'on trouve le mot « dégriser ». On s'aperçoit alors que le mot « écrire » se trouve dans le quart compris entre « dégriser » et « invincible ». Il ne reste qu'à répéter à nouveau l'opération de séparation au milieu de ce quart et ainsi de suite jusqu'à tomber sur la page contenant le mot « écrire ». En termes mathématiques, cette

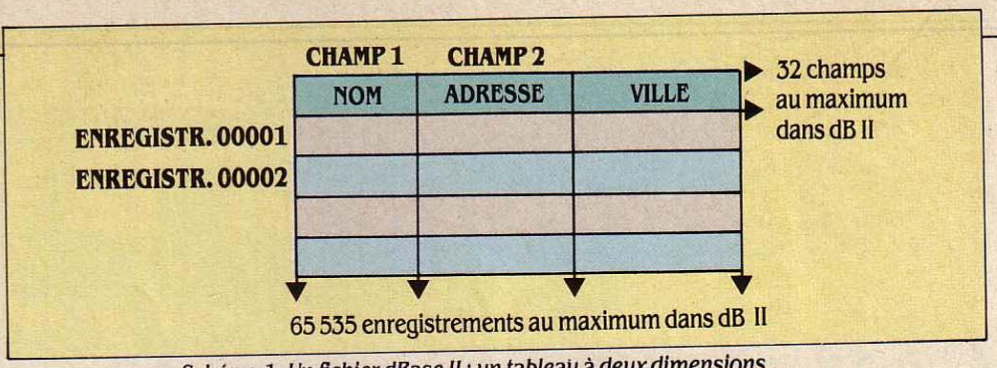


Schéma 1. Un fichier dBase II : un tableau à deux dimensions.

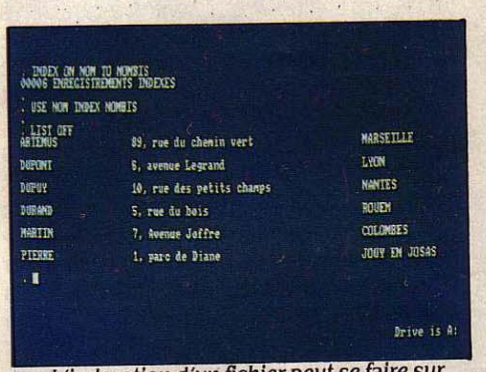
EN-TETE D'UN FICHIER dBASE II

OCTET	CONTENU	SIGNIFICATION
0	02H	Identifie un fichier dB II
1-2	Nombre codé sur 16 bits	Nombre d'enregistrements dans le fichier
3-5	3 octets	Date de la dernière mise à jour
6-7	Nombre codé sur 16 bits	Taille de l'enregistrement
8-519	Tableau de 16 x 32 octets	Indicateur de la nature des champs
520	1 octet	Prend la valeur ODH si les 32 champs sont définis

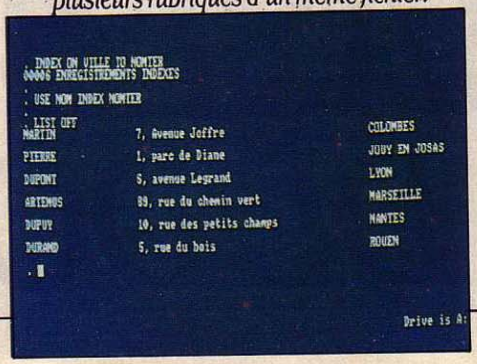
STRUCTURE DES CHAMPS

OCTET	CONTENU	SIGNIFICATION
0-10	11 octets	Nom du champ en ASCII
11	1 octet	Type du champ en ASCII (C, N ou L)
12	1 octet	Longueur du champ
13-14	Numéro codé sur 16 bits	Adresse du champ
15	1 octet	Compteur de champs en décimal

Schéma 2. La structure d'un en-tête d'un fichier dBase II. Des informations accessibles à tout moment par la commande DISPLAY STRUCTURE.



L'indexation d'un fichier peut se faire sur plusieurs rubriques d'un même fichier.



recherche nécessite (n+1)/2 opérations (où n représente le nombre d'enregistrements dans une base de données) et fournit un moyen extrêmement rapide de rechercher un enregistrement dans une base. Cette méthode présente cependant le grand inconvénient d'être coûteuse à gérer lorsque l'on rajoute un enregistrement : il faut en effet retrier le fichier tout entier pour remettre le nouvel enregistrement à la bonne place, ce qui prend un temps proportionnel à la taille des fichiers. Une autre méthode consiste à « hacher » les données du fichier. On crée alors une « table de hachage » (de l'anglais hashing) que l'on indexe d'une façon simple. Si l'on reprend l'exemple du dictionnaire, il suffit de réserver une case pour les mots commençant par A, une autre pour B, etc.

Architecture en arbre

L'inconvénient de cette méthode est que l'on perd énormément de mémoire. En effet, à moins de savoir au préalable le nombre d'enregistrements qui viendront s'inscrire dans la table, il faut réserver suffisamment d'espace mémoire pour chaque lettre. Ainsi, en français, les mots commençant par A, B, E, etc., seront nombreux, mais ceux commençant par Y, Z, W le seront beaucoup moins, d'où un gâchis énorme de mémoire. La troisième technique est celle de l'architecture en arbre. Cette méthode est appelée ainsi car la façon dont sont organisées les données représente à proprement parler un arbre inversé (voir schéma 3). Le premier enregistrement, (ici Bertrand) forme le point le plus élevé, encore appelé la « racine ». Les suivants s'articulent vers le bas autour de lui et constituent les « branches ». De la racine partent toujours deux et seulement deux branches. L'intersection d'où partent les deux branches s'appelle un « nœud ». On se réfère aussi souvent à un arbre généalogique dans la mesure où chaque « parent » a exactement deux « enfants ». L'organisation d'un tel arbre est telle que si la donnée recherchée est égale à la valeur du « nœud », l'enregistrement est trouvé ; si la donnée est plus grande que la valeur du nœud, la recherche est poursuivie vers la branche de droite ; si la donnée est plus petite, on cherche vers la branche de gauche. Ce genre de structure permet de retrouver un enregistrement très rapidement. Une recherche s'effectue en fait sur le mode binaire : il n'y a que deux choix possibles, aller vers la droite, ou aller vers la gauche. Ainsi pour rechercher un enregistrement parmi 1 024 (soit 2^{10}), il suffit de 10 comparaisons. Une base contenant 1 048 576 enregistrements (2^{20}) ne nécessitera que 20 comparaisons. Cette structure en arbre binaire serait idéale si tous les enregistrements se trouvaient en mémoire vive. Mais dans la plupart des cas – et dans dBase II notamment –, les données sont conservées sur la disquette.

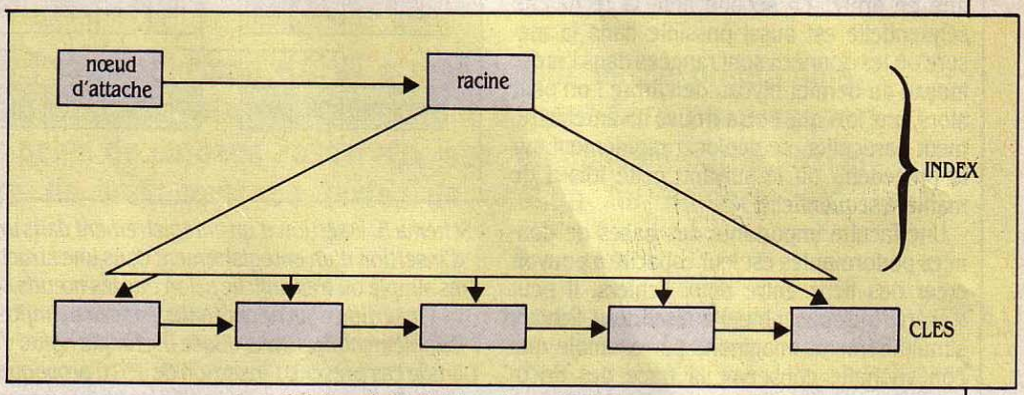


Schéma 4. Un fichier séquentiel indexé : les données sont accessibles à la fois directement et séquentiellement. Seul le dernier niveau (celui des feuilles) contient des clés qui pointent vers les enregistrements.

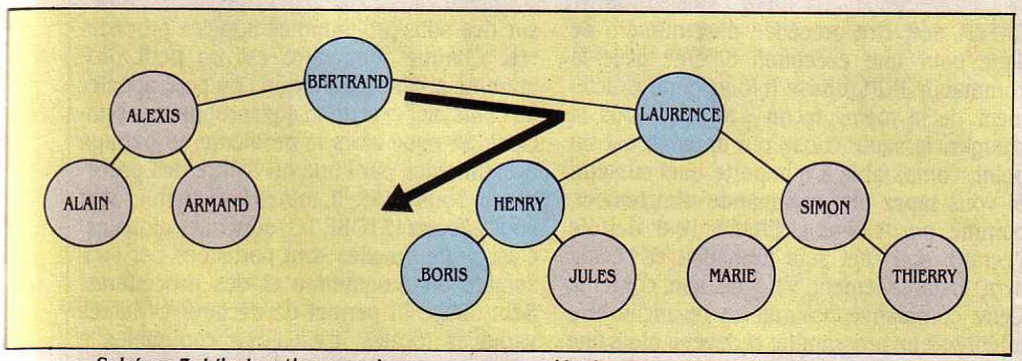


Schéma 3. L'indexation en arbre suppose que si la donnée recherchée est plus grande que la valeur du nœud, la recherche est poursuivie vers la branche de droite et si la valeur est plus petite, vers la branche de gauche. Ici, pour rechercher le nom « Boris », tout le cheminement indiqué en couleur doit être parcouru.

Pour retrouver une information, il faut parcourir une partie des enregistrements de l'index à partir d'un nœud. Si ces enregistrements étaient physiquement sur la disquette, les uns derrière les autres, on minimiserait ainsi les déplacements de la tête de lecture et on diminuerait d'autant les temps de recherche. D'où la préoccupation des concepteurs de logiciels de gestion de bases de données à favoriser des structures d'index qui génèrent des lectures séquentielles, ce qui n'est pas le cas de la structure de l'arbre binaire. En effet, si l'on reprend l'exemple du dictionnaire et de la structure en arbre, on s'aperçoit qu'il est très facile et rapide de retrouver un mot mais pas nécessairement celui qui se trouve juste après ou juste avant : il peut en effet être indispensable de repartir de la racine pour rechercher les mots voisins. Or on a très souvent besoin lorsqu'on manipule une base de données, de retrouver facilement et rapidement un ou plusieurs éléments adjacents. Pour remédier à cela, on a imaginé une version améliorée de la structure en arbre : celle dite du « B+arbre ». Dans une structure en arbre classique, chaque nœud contient une adresse associée à un enregistrement particulier. C'est ainsi que lorsque l'on recherche un

LES LIMITES THEORIQUES DE DBASE II

MALGRÉ DES CAPACITÉS ÉTENdues, dBase II n'en demeure pas moins limité. Le premier reproche porte sur sa rigidité dans la structure des données. Une rubrique définie au départ pour accueillir 20 caractères consommera quoiqu'elle contienne, 20 caractères. Cette rigidité impose que l'utilisateur connaisse a priori la longueur approximative des différentes rubriques. On rencontre cependant ce genre d'inconvénient, lié aux enregistrements de longueur fixe sur beaucoup de systèmes de gestion de base de données. La seconde limite importante vient de la restriction du nombre de fichiers pouvant être créés. dBase II ne peut, par exemple, gérer plus de sept fichiers d'index. Cette limite peut d'ailleurs être encore réduite en fonction de l'environnement physique sur lequel il est utilisé. Le nombre de variables mémoire est limité à 64, chacune ayant une longueur maximum de 254 caractères avec de toute façon un maximum de 1536 caractères pour toutes les variables. Le nombre de caractères maximum par rubrique ne peut dépasser 254 ; quant au nombre de rubriques définissables dans un fichier, il doit rester en-dessous de la barre des 32. Enfin, le nombre maximum d'enregistrements par fichier ne peut exéder 65 535, ce qui peut être un sérieux inconvénient dans une utilisation professionnelle. En dernier lieu, c'est sur le nombre de fichiers pouvant être ouverts simultanément que dBase II montre sa faiblesse. Alors que dBase III permet d'ouvrir jusqu'à dix fichiers simultanément, son prédécesseur n'en autorise que deux. Naturellement, la place disponible dans la mémoire de l'Amstrad et sur sa disquette est une contrainte qui vient s'ajouter à ces chiffres théoriques.

enregistrement et qu'une correspondance est trouvée dans l'index, l'enregistrement correspondant est immédiatement disponible. Dans le B+arbre, seules les « feuilles », situées au dernier niveau de l'arbre, contiennent les adresses qui renvoient aux enregistrements. Les nœuds ne contiennent que des données permettant de s'orienter tout au long de l'arbre jusqu'au dernier niveau. Cette méthode est dite « séquentielle indexée ». Cela présente plusieurs avantages non négligeables : en premier lieu, on conserve le bénéfice de la structure en arbre. En second lieu, la recherche séquentielle est aussi possible dans la mesure où les données sont rangées dans l'ordre indexé au dernier niveau de l'arbre : on peut alors, une fois que l'on a trouvé un enregistrement particulier, se déplacer rapidement sur le précédent, ou le suivant, cette fois-ci de manière séquentielle.

Une faculté importante des bases de données performantes est leur capacité à pouvoir créer des liens entre deux fichiers. Il peut s'avérer indispensable d'utiliser deux fichiers simultanément. Imaginons par exemple que l'on souhaite conserver la trace des documents d'une entreprise grâce à dBase II. A chaque fois qu'un client passera une commande, il va falloir saisir non seulement les caractéristiques de la commande qu'il vient de passer (nature de l'article, quantités, montants, etc.), mais aussi les informations propres au client (nom, adresse, solvabilité, etc.). Il en résulte une redondance importante. Il serait plus simple de garder ces dernières informations dans un fichier séparé. Si on pouvait relier le fichier « commandes » et le fichier « clients » par l'élément qui leur est commun, (c'est-à-dire le nom du client), il suffirait d'indiquer à l'ordinateur le nom du client, et ses coordonnées seraient prises en compte automatiquement. dBase II fait cela.

Il permet de réserver au maximum deux espaces mémoire pour manipuler deux fichiers simultanément. Les deux espaces sont appelés PRIMARY et SECONDARY et peuvent être commutés sans qu'un des deux fichiers ne soit refermé. Pour relier les deux fichiers dans l'exemple précédent, la méthode consiste à sélectionner le fichier clients comme étant dans la zone PRIMARY, et le fichier commandes dans la zone SECONDARY. Grâce à ces deux fichiers, on peut obtenir les coordonnées d'un client ainsi que le chiffre d'affaires réalisé avec lui jusqu'à présent. La méthode consiste à relier les deux fichiers par leur champ commun qui est en l'occurrence le nom. Vous pouvez alors sélectionner un nom dans le premier fichier, le garder en mémoire, ouvrir le second fichier et rechercher toutes les fois où ce nom apparaît dans les commandes. Bien entendu cette recherche peut être automatisée grâce au langage de programmation évolué dont dispose dBase II.

Comme dans la majorité des langages de programmation, il existe deux modes qui permettent d'exécuter des ordres : le mode direct et le mode programme. Les familiers du Basic savent que l'instruction PRINT peut soit être frappée directement et exécutée grâce à ENTER, soit être précédée d'un numéro de ligne pour une exécution différée avec la commande RUN. dBase II fonctionne exactement de la même façon ; lorsque vous le chargez, la seule chose qui apparaît est un point, comparable à n'importe quel curseur. Si vous tapez une commande directement, comme par exemple ERASE (qui nettoie l'écran), son effet sera immédiat et l'écran sera instantanément vidé de son contenu. Cette commande pourrait en revanche être intégrée à un programme et devenir alors une instruction, dont l'action ne prendrait effet qu'après avoir lancé le programme (commande DO suivie du nom du programme).

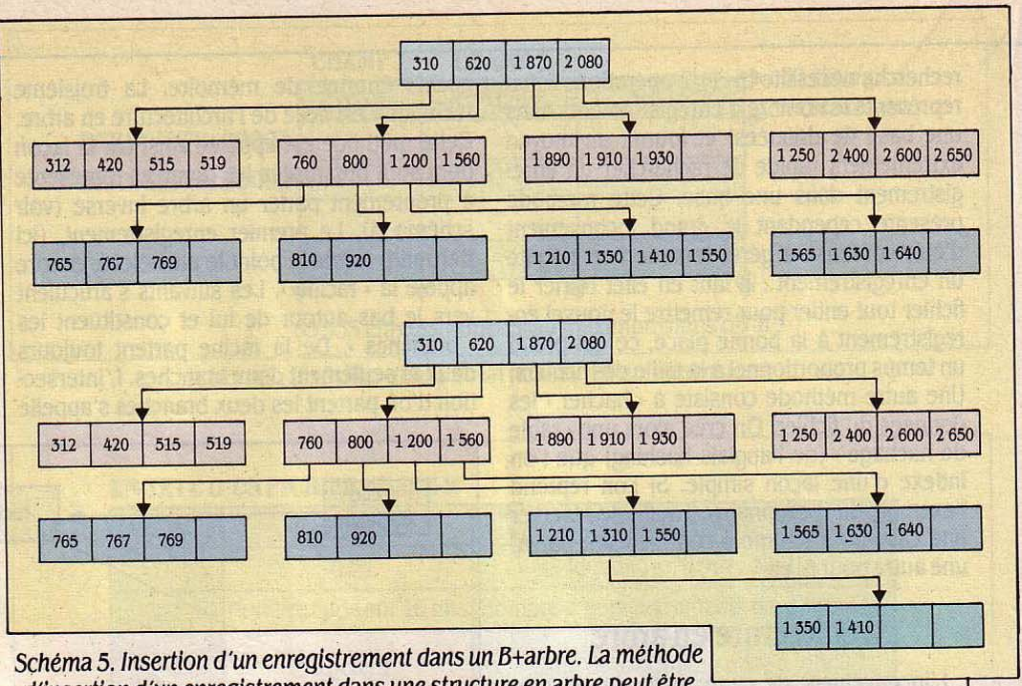


Schéma 5. Insertion d'un enregistrement dans un B+arbre. La méthode d'insertion d'un enregistrement dans une structure en arbre peut être très simple ou très difficile selon que les nœuds sont pleins ou non. Ce schéma montre un cas où l'insertion d'un nouvel enregistrement impose que l'on divise l'arbre en une sous-racine supplémentaire. Dans dBase II, chaque nœud de l'arbre possède une taille fixe de 512 octets. Dans le cas présent, l'insertion de 1310 provoque la création d'un nouveau nœud (couleur verte). Cette scission a pour effet de libérer un espace libre à côté de 1550 ; si le prochain enregistrement prend une valeur comprise entre 1550 et 1560, il pourra s'y glisser sans qu'il y ait création d'un nouveau nœud. L'insertion de cette nouvelle valeur ne modifie pas la hiérarchie de l'organisation des données.

Pour programmeurs avertis

A l'inverse de certains langages, il n'y a pas de numéros de lignes sous dBase II. Les programmes se construisent à partir d'un éditeur intégré (comme on en trouve d'ailleurs de plus en plus sur des langages de programmation classiques) qui permet d'écrire les lignes, de les supprimer, de les modifier, etc. Un programme écrit pour dBase II est, en fait, en tout point semblable à un document écrit sur un traitement de texte. Un programme dBase II est tout simplement une suite d'instructions que le logiciel va exécuter séquentiellement. On y retrouve beaucoup d'instructions que les programmeurs en Pascal connaissent : CASE OF, qui permet de se brancher sur une routine selon les cas, DO WHILE, un peu similaire au FOR...NEXT du Basic, mais dont l'argument après le WHILE peut être modifié en cours de traitement. Il n'y a pas de GOTO et encore moins de GOSUB sous dBase II, dans la mesure où l'on ne peut se brancher vers des numéros de ligne, mais sur des sous-programmes appelés procédures. Chaque procédure est un petit programme à part entière que l'on peut appeler au cours de l'exécution d'un programme principal. Se pose alors le problème du partage des variables par l'une ou l'autre des procédures. Sous dBase II, toutes les variables que vous affectez (STORE TO) sont dites globales, c'est-à-dire qu'elles sont partagées par l'ensemble du programme et des procédures. Seul dBase III permet de déclarer certaines variables locales, c'est-à-dire communes à une seule procédure. Pour simuler une variable locale en dBase II, vous avez la possibilité de définir une variable en mémoire, d'appeler votre procédure, puis de la supprimer grâce à la commande (ou instruction) RELEASE suivie du nom de la variable.

Fonctionnellement, dBase II est identique sur Amstrad et sur les ordinateurs professionnels. La puissance dont on dispose est donc réelle. Mais il ne faut pas perdre de vue que les caractéristiques de l'ordinateur imposent des limites à la taille des fichiers que l'on veut manipuler : les disquettes ne stockent que 160 Ko sur chaque face, et il n'y a naturellement pas de disque dur. Le couple dBase II-Amstrad paraît donc bien adapté à deux utilisations : côté loisirs, l'initiation aux bases de données et à la programmation en dBase II, où il offre un champ d'exploration considérable ; côté professionnel, la gestion de petites entreprises limitées à une centaine de clients ou de fournisseurs, par exemple. Dans ce dernier cas, il ne faut pas oublier que la « construction » d'un fichier avec dBase II n'est pas une opération facile : inutile de l'envisager si vous n'avez pas un goût certain pour la programmation.

Eric TENIN