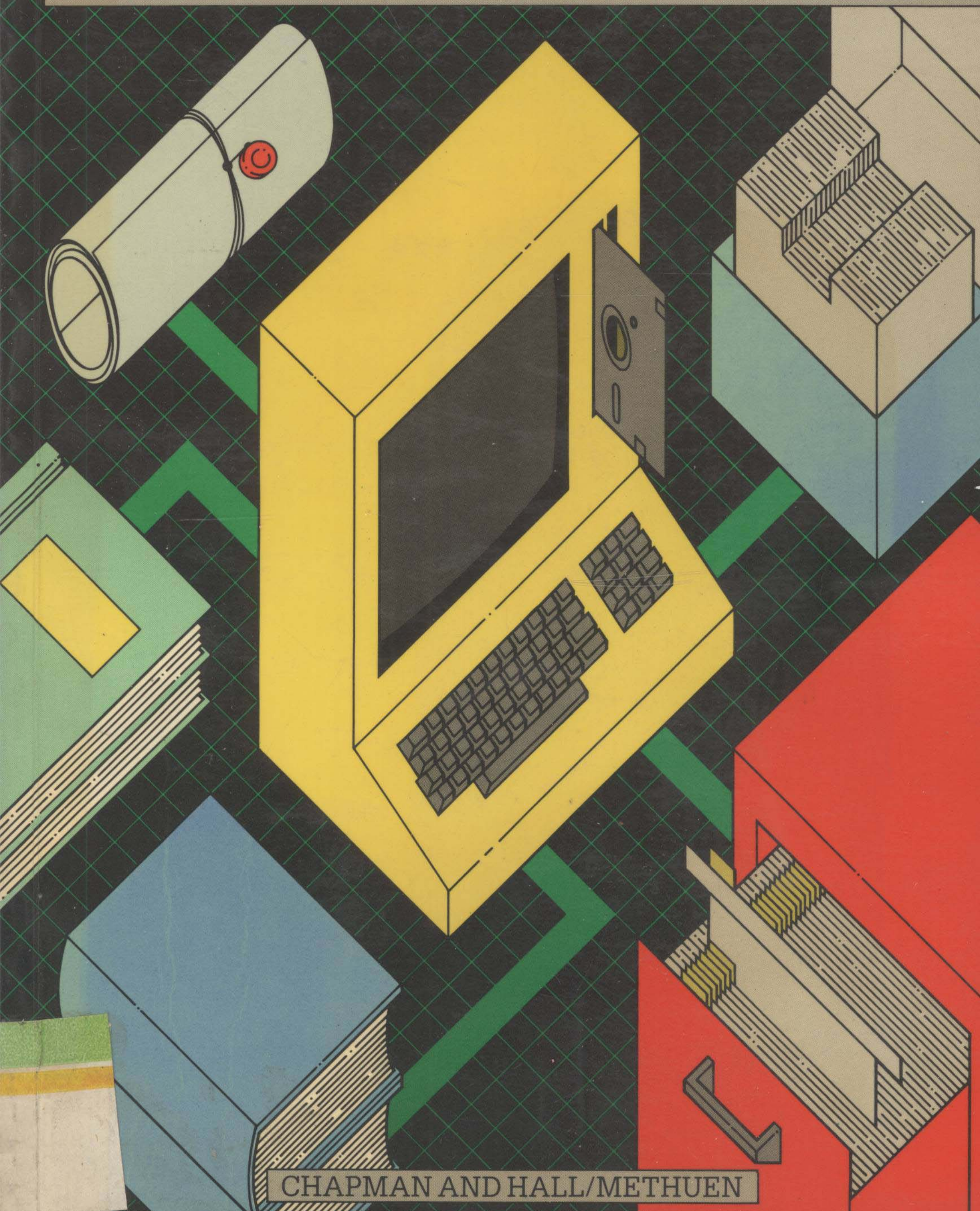
PETER LAURIE

DATABASES

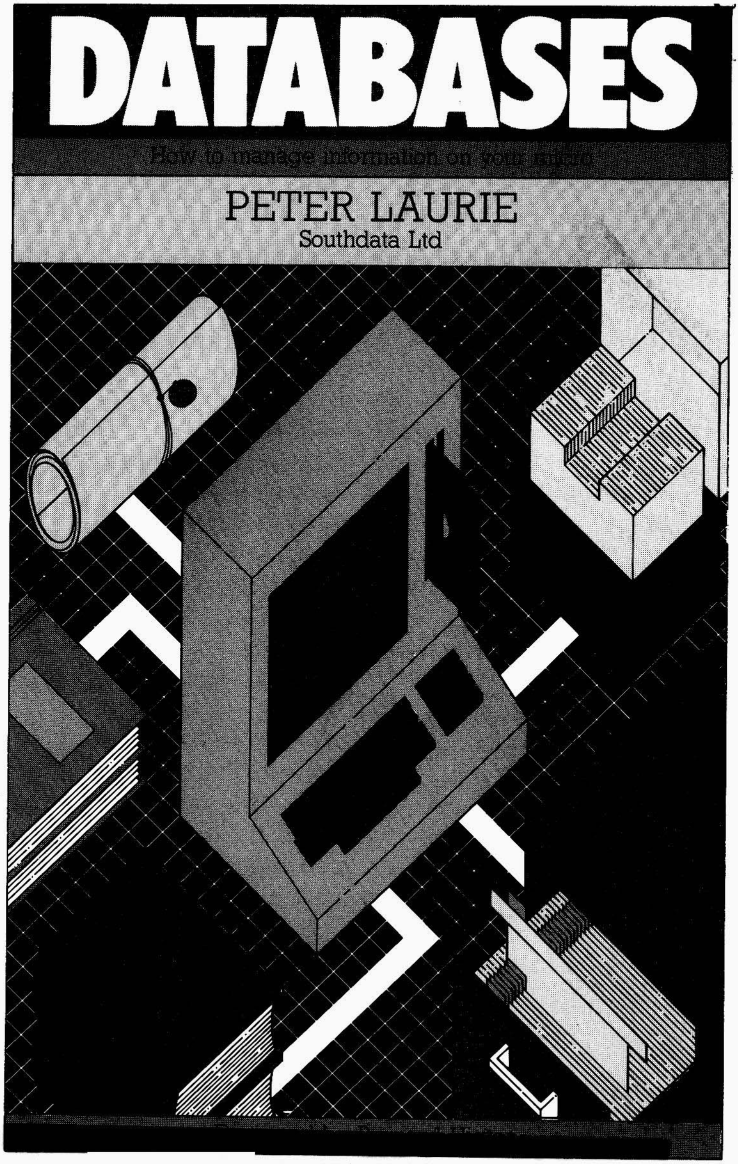
How to manage information on your micro



CHAPMAN AND HALL/METHUEN

DATABASES

How to manage information on your micro



CHAPMAN AND HALL/METHUEN
London New York

First published in 1985 by
Chapman and Hall Ltd/Methuen London Ltd
11 New Fetter Lane, London EC49 4EE

Published in the USA by
Chapman and Hall/Methuen Inc
733 Third Avenue, New York NY 10017

Diagrams by Trevor Bounford
Illustrations by Val Hill

Printed in Great Britain at the
University Press, Cambridge

ISBN 0 412 26380 7

Laurie, Peter
Databases: how to manage information on your micro.
1. Data base management 2. Microcomputers
I. Title
001.64'42 QA76.9.D3

ISBN 0-412-26380-7 Pbk

Library of Congress Cataloging in Publication Data

Laurie, Peter.
Databases : how to manage information to your micro.

Bibliography: p.
Includes index.
1. Data base management. 2. Microcomputers
Programming. I. Title.
QA76.9.D3L38 1985 001.64'2 84-22976
ISBN 0-412-26380-7 (pbk.)

CONTENTS

PREFACE

This book is an introduction to the basic ideas of database management. This may seem an unnecessarily esoteric subject until one reflects that very few offices, and indeed very few homes, do not contain a card index or a filing cabinet, and that if the information they contain were transferred to a computer it would be – by definition – a database. If one believes that within five years every office worker will have a personal microcomputer, then the corollary is inescapable: we shall all have to understand database management.

Amateurs in the computing field should find this a challenging but rewarding book. Because of the huge market for microcomputers and the equally huge lack of knowledge about them, there is a vast premium at the moment on 'user-friendly' software which purports to work without any help from the user. The current crop of easy-to-use database software often does not, in my view, address the real issues in information management. There is great pressure to invest programming effort in gimmicks that will make the naive and potential purchaser of the equipment believe that he or she does not have to understand how the machine works. To change the analogy somewhat, today's micros are rather like the motor cars of 1910. You *could* drive to Peking (or Beijing) in them, but you had to be prepared to regrind the cylinder block by hand somewhere back of Lhasa. If you went motoring you soon became a competent mechanic. By the same token, if you are not computerate when you begin a database project, you certainly will be by the time you bring it to a happy conclusion. As Kutuzov, the Russian general who opposed Napoleon's invasion, used to tell the chaps, 'Train hard, fight easy.'

Professionals in the computing field may find this an intriguing but rather unorthodox book. Until recently, database management has been an important but small and remote speciality. Few people worked in it and they have developed a theory of database management at a high level of abstract discussion. At times the theory seems almost ritual in its opacity and irrelevance to what appear to me to be the real needs of users. When, in 1980, I needed to find out about this subject because my company was starting to write and publish a database package*, I found surprisingly little that was useful in the existing texts.

We wrote the software by the light of nature – and very lucky we did too (see Chapter 4). My knowledge of what databases ought to do started from

*The package is called *Superfile* in the UK and Europe, *Avatar* in the US; the company Southdata Ltd.

talking over the last three years to some thousands of people who wanted to manage information on micros. These conversations led to endless debates with the programmers – headed by my son Bennet, without whom there would be no Southdata, no Superfile and no book. By dint of a lot of talking, arm-waving, programming and trying the results out on real, live customers we slowly arrived at what seems to us to be a workable product. Out of it too has come this book which will, I hope, illuminate some puzzling corners of an important subject.

I must apologize in advance for letting my own business preoccupations intrude into the argument. An author is supposed to be impartial; to have no commercial axe to grind. Unfortunately the world of computing is so young and so small and expanding so rapidly that anyone with any ideas is expressing them in as many ways as possible. It is like the Wild West: the sheriff who prosecutes you for cattle rustling often turns out to own the largest ranch in the territory. It is unfortunate but there are no innocent bystanders in computing either.

It would have been nice if this book could have served as a catalogue of currently available products, with their features described and an assessment of performance given. Unfortunately there are now just too many database packages; at least 60 are available in the UK alone and in the US there are more than 150 competing database software packages. To review a piece of software properly requires about a month's work, so the current offerings would take more than 17 years, by which time it will all be irrelevant.

This book is not intended as a handbook for software professionals. A glance at the 'bible' – Donald Knuth's *The Art of Computer Programming*, (1973), in seven volumes – will explain why no book this size could convey anything useful. The aim of this work is to outline the salient issues and techniques to the intelligent amateur, and still partly perplexed professional so that he or she stands some chance of picking a way through the jungle of competitive products.

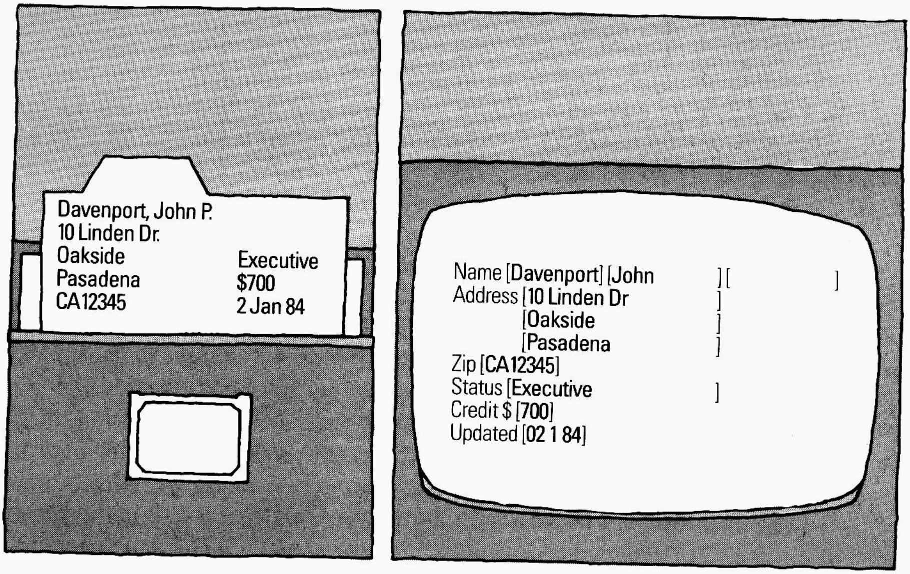
CHAPTER ONE

THE COMPUTER

A database is a bunch of information held in a computer's memory. It is useful because one can look for things in it not just by the file number or the addressee's name, as with a paper filing system, but by almost anything that is in it. But before we talk about the trials and tribulations – and also the convenience and uses – of database management, it would be a good idea to digress here to talk about the salient features of the computer itself as it affects the user of a database system.

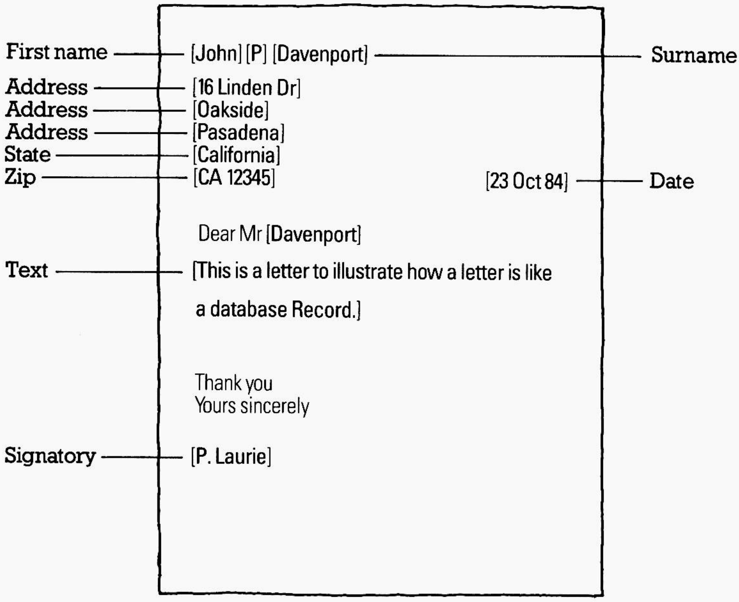
1.1 The introspective typewriter

One can – and many writers have – become quite philosophical about computers. For the present purpose, that is in considering how they can be used for the storage and retrieval of information in databases, a computer is just an electric typewriter (with a certain amount of introspection) built into a filing cabinet. By introspection I mean that you can ask the typewriter to look at what it has typed – not only recently, but two years ago – and ask it to, 'find me someone called "Tom" or anyone who owes me £500'.



A 'Record' in a computer database (right) is much the same as the information one might write on an index card (left). Each item of information on the card goes into a labelled 'field' on the computer screen.

Information in a database is assumed (for the moment, but see Chapter 4) to consist of numbers of similarly shaped chunks of data. Each chunk is called a 'record'. A record is made up of 'items' which, in turn consist of 'fields' and 'values'. A field is a space for a particular type of information – a surname or the date a letter was written or the price paid for an object. A letter in a filing cabinet can be broken down into these components and would make a very suitable record in a free-text database. We will look more closely in later chapters at the way such things are organized.



The elements of a business letter can easily be seen as fields in a database Record.

A database consists of a lot of chunks of similar data held on a disk. Database management is all about memory and so, what the computer has to offer in this department is very important. The machine has two sorts of memory: its internal Random Access Memory (RAM) which is connected directly to the processor and its 'external' disk memory. External is in quotation marks because, although the disks may be built into the same box as the computer, they are logically and functionally exterior to it in the same way that the keyboard and screen are.

The RAM available in desk-top micros is relatively small – between 64K*

*1K = 1024 bytes. A byte is the unit of information storage and holds one letter or number.

and 500K. When the power is turned off the contents of RAM generally disappear. The disk holds data permanently (one hopes) and is much bigger. In modern desk-top micros you would expect to have about 1MB (million bytes) on a floppy disk and up to 20MB on hard disk. If you want to spend the money you can have many thousands of megabytes of disk storage. And, as we shall see later, devices are coming soon which will provide these huge storage capacities using cheap, new technologies.

One reckons that a text word is on average six characters long, so 1MB holds 166 666 words or about three ordinary sized novels. A good typist will rattle the keys at 80 words per minute – it would take him or her 33 non-stop hours to fill up a megabyte. But it is surprising how quickly even as much space as this disappears. The machine on which I wrote this has 16MB of disk and 15MB of it was full by the end. We could always erase the files that do not seem to be relevant, but, as sure as God takes bytes out of little green magnetic apples, that file will be the pride of someone's eye. It seems that most of the expensive hard disks in the world are three quarters full of files that no one can remember a thing about. A variant of Parkinson's Law – data expands to fill the storage available – applies very strongly. The new high capacity disks promise enough storage to hold a lifetime's work from a fast typist, which means they will take only a year to fill up.

The computer cannot do anything directly with data on disk. It has to be copied into RAM before it can be searched or arithmetic can be done on it. RAM is rather like your desk top; the disk is like the filing cabinet. You get selected papers out of the filing cabinet, work at them on your desk and put them back. However, if you are using a computer with a database manager, you have the assistance of a very fast and intelligent clerk who will find the papers you want from pretty vague hints about what it is in them.

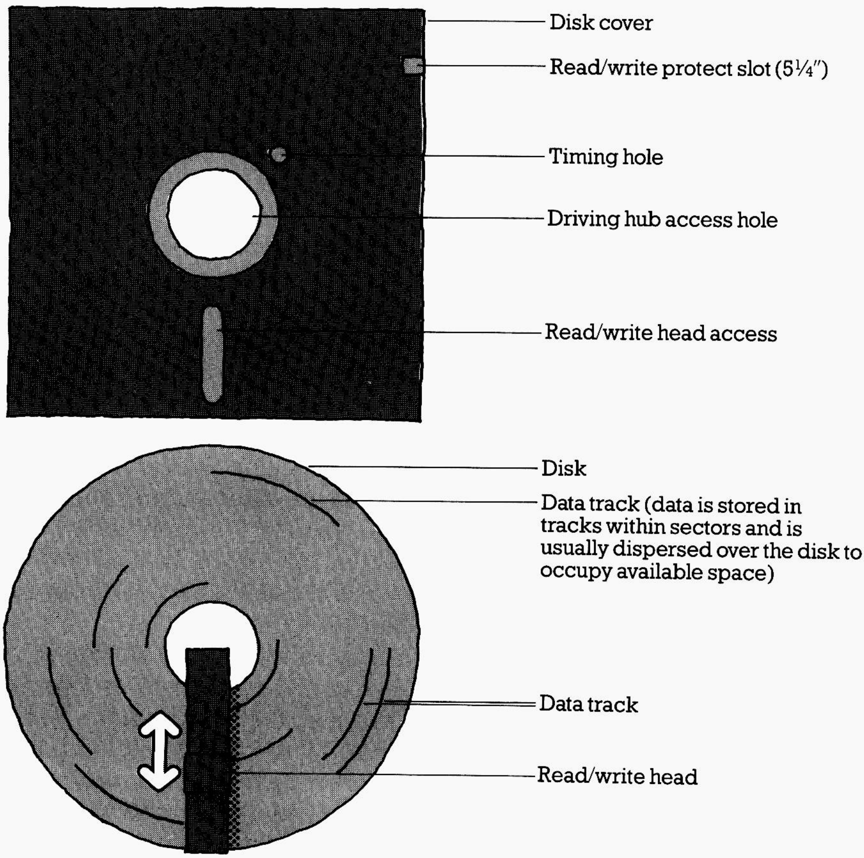
1.2 Disks and Data

Data is stored on a disk – whether floppy or hard – as minute patches of magnetism. The patches go north to south to record a '1', or south to north to record a '0'. (Actually, as always, it is more complicated than that, but this is the general idea.) The 0s and 1s are called 'bits'.

The patches of magnetism are written into the magnetic material that covers the disk by the 'read-write' head. This is a tiny electro-magnet that skims the surface of the disk. Apart from the fact that it and the magnetic material both move, the principle is just the same as in a tape recorder where the head stands still and the magnetic material alone moves past it.

The same head† serves both to write new patches of magnetism and to

†There may actually be several heads and several disks or 'platters', but the principle is the same.



A floppy disk (above) rotates inside its cover (top) to bring data written on it under the read-write head. A hard or 'Winchester' disk works the same way except that the disk surface cannot be removed from the machine.

read old ones. The disk spins round and the head moves in and out so that every part of the disk surface can come under the head by a combination of these two movements. The head writes data in a series of concentric 'tracks' which are broken up into 'sectors'.

The operation that takes the time is positioning the head over the right track on the disk and waiting for the right sector to come round. This takes roughly a fifth of a second on a reasonable floppy disk machine. A hard disk takes about 1/300 of a second, but it has to swim its head about over much more data so the rate at which it can search in bits per second is perhaps only ten times higher. In estimating the amount of time that a database manager will take to do anything, you have to look at the number of disk accesses involved.

As must be well known in the western world by now, the bits represented by the little patches of magnetism of the disk are made up into 'bytes'. Each byte consists of eight bits; 1s or 0s. Since each bit can have two values – 0 or 1 – a byte can have $2^8 = 256$ different values, which is plenty for upper and lower case letters, numbers, punctuation, brackets and all the other characters one finds on a keyboard, plus some more.

A file consists of a string of bytes. What else could it be? The computer can interpret these bytes in several ways: as a command to its processor which means the file is a program; as letters and numbers, or in many other ways. Since databases are about human-type information, we want to look at the byte as a letter or a number – i.e. as the equivalent of a key on a computer keyboard. The generally used code is the American Standard Code for Information Interchange (ASCII) which assigns a number from 0 to 127 to each keyboard character. Upper and lower case letters have distinct codes and are treated as different characters. 'A' is ASCII 65, 'a' is 97. There are some 'non-printing' characters like Line Feed, Carriage Return etc. along with 32 invisible characters made by pressing the Control key with a letter. These are used by some packages to allow the user to give commands. In a word processor, for instance, Control F might mean move the cursor forward one character.

ASCII (as its name implies) is fine for Americans, but imposes a very annoying kind of cultural colonialism on its users. The difficulty arises from the representation of language-specific characters like é in Spanish or ç in French. ASCII assigns codes to these letters but it assumes that a user wants either Spanish or French but never both at once. Thus, for instance, ASCII code 43 is the hash sign in America or the pound in Britain but not both. If, in Britain you want hash you can't have pound – and the reverse in America. The code for ']' in English or American is ä in Swedish and ü in German. The upshot is that you cannot keep a database with words in it from more than one European language. It is all very annoying. No doubt the Americans who invented ASCII found it weird enough that foreigners would want to spell in one foreign language let alone two or all of them.

The person who drew this problem to my attention was an Englishman who wanted to keep a database of military publications. His authors came from all nations under the sun, but he found that if he were to use a computer he could not spell their names properly. No doubt the micro revolution will purge European languages of their idiosyncratic characters.

If you work in Arabic it is possible to assign byte codes to the characters of your alphabet. If Chinese or Japanese is your poison, you can give each character two bytes and so cope with 256^2, or 65 536 of them.

However, what we want most of the time is to store letters and numbers from our own language and this can be done very conveniently in the form of bytes. The computer will keep them in a file on the disk. In small machine operating systems, like MS-DOS and CP/M, a disk file is a very definite thing: a long string of bytes. It has a name and a length. The database manager will keep its information in one or more data files. It will almost certainly keep indexes to the data in separate files and may well keep other little bits of information in other files still.

Not only data gets kept as files. The machine will keep its programs in the same form, and those programs will need subsidiary files. If you use the directory command on your machine you will see a great mass of files with names of the form 'xxxxxxxx.yyy'. The 'xxx' part is the filename, and 'yyy' the extension. The extension tells the user and the computer what sort of file it is. Programs usually have '.COM' or '.CMD' extensions because they can command the computer. Text files often have '.TXT' extensions, databases '.DAT' and so on. The extension is useful because you can make the machine display, copy, print or whatever all files of the same type. For instance the chapters of this book consist of a set of files on my computer called: C1.B, C2.B, C3.B,. C8.B.

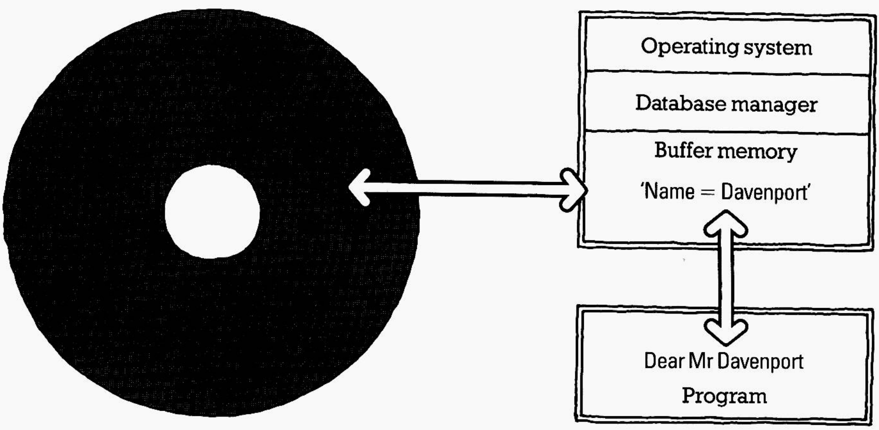
The '.B' extension means I can back up the whole book with a single command: 'COPY *.B B:' (copy all the files with the extension .B to disk drive B).

The database manager gets at the files through the 'operating system'. This is a piece of software that lives in the computer and does all its basic housekeeping tasks, like getting files on and off the disk, getting characters from the keyboard, printing on the screen and the printer.

1.3 The operating system

The operating system has an important and quite invisible job to do. It is faced with a tricky problem. When you start out with a new computer and an empty disk, things are fine. You write your first file onto it, starting at the beginning, and from the end of that you write the second. You carry on until the disk is full up and then you stop. There is not a bit of wasted room. But, life being what it is, full of change and bustle, you will have altered the first file. It will now be either longer or shorter than it was originally. If it is shorter there is now an unused space. If it is longer, where do you put the extra chunk? It can't go at the end of the first file because that is where the second file starts.

The answer is to split the disk up (notionally) like a tesselated floor. It is divided into small chunks of between 128 and 1024 bytes each. Each chunk is used to store a piece of the file. (These chunks are, confusingly, called 'records'. We shall spell them with a small 'r' to distinguish them from the human-type units of information – Records – that live in the database. We will meet them in the next chapter.) Whenever a file gets shorter or is erased, some chunks become free somewhere on the disk and can be used for the longer bits of other files. The operating system keeps its own disk directory which tells it what chunks are used in what order by each file. When the program running at the time wants to read a file the operating system automatically gets the records off the disk in the right order and puts them together to present a seamless whole. If a program wants to read only part of a file, it asks the operating system to give it record number so and so from such and such a file. If the program is a database manager and the file the database, it will get the appropriate record number from its indexes (Chapter 2). The operating system will copy the record into a 'buffer' in memory where the program can get at it. A buffer is an area of RAM which can be used as a notice-board, where one program can pin up messages for another. Generally, the bigger these buffers, the better everything works because the number of disk accesses to do any particular job is reduced. But, of course, the space given to buffers has to compete with space allocated to program, so the designer of a database manager has to compromise.



Raw data from the disk is passed through the operating system to the database manager. It can then be accessed by the applications program – here printing a letter.

Programs are said to 'run under' an operating system. Those in common use are: CP/M 80 (for 8 bit machines) , CP/M 86 (for 16 bit machines), MS-DOS, Unix, Idris and Xenix (for 16 bit and bigger machines only) . A program like a database manager will use the operating system to get at its data files. It will be able to ask for a whole file, or maybe just the record that starts at location 100 218 on the disk. The database manager gets the number 100 218 out of its indexes, which are themselves disk files.

A very important quality of the database manager is the speed with which it finds anything. Programmers who write database packages spend a great deal of time and effort trying to maximize this speed. It is as well for the reader, as a possible customer for such systems, to understand what the constraints are on fast searching.

The records are distributed among the tracks and sectors, but only the programmers who wrote the operating system have to worry about that. To find any particular piece of data, the operating system is given the number of a record in a particular file. It looks up in the directory to see where that record is physically on the disk and then moves the head to the right track. It then has to wait for the disk to spin the right sector under it. The head then reads the data off the disk into a buffer in the computer's RAM. The database manager will copy it into its own buffers and maybe ask for the next record if the data Record is larger (as it almost certainly will be) than the operating system record.

The amount of data that can be kept by a database manager depends on the amount of disk space available to be shared between its files. This disk space depends on: (1) the physical size of the disks – and that depends largely on the depth of the purchaser's pocket; and (2) on the largest disk size the operating system will handle. For instance, 8 bit CP/M only allows 8MB.

The maximum practical database will almost certainly be a lot smaller than the available disk space because the database manager has to maintain indexes and it may well use fixed length records which waste a lot of space in empty fields.

1.4 Peripherals

The computer is just a box, a black box if you bought a black computer, otherwise the colour of the manufacturer's choice. (Naive people often think that the shape and colour of the box are important. They say 'Oh, what a pretty computer!' and wonder why programmers spit on the floor. As with Chinese restaurants, good decor is no indicator of good fare.)

If you are lucky the box will have a light or two on the outside to show that it is alive, but in other respects it is not very amusing. What makes it usable is its peripherals: the keyboard and screen (known collectively as the 'console'), the printer and (more rarely) other things like graph plotters. Most desk-top computers are supplied with a screen and keyboard, but if you have a larger multi-user micro you will have to buy separate VDUs (visual display units) .

Even the smallest computers now run database managers. Although Sinclair QL's tape-loop data storage devices are too slow and small for proper database work, it comes with a database package. Here it shows a sample database with worldwide economic data.