# EDUCATIONAL DATABASE MANAGEMENT SYSTEMS

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF MATHEMATICS

COMPUTER SCIENCE

Master of Philosophy

# EDUCATIONAL DATABASE MANAGEMENT SYSTEMS

by Stephen Hitchman

Computer education, particularly in schools and colleges, has been the focus of much criticism. This thesis discusses the possibilities and implications of using a database management package as the focus for the core of any computing course.

The thesis outlines data and database fundamentals, the current state of computer education, the tools available for computer studies teaching, a survey of business user requirements from the point of view of employee database management skills, and a strategy for using a database management package for a computer studies course.

The author finds significant benefits for students who learn computing through a database package. This strategy enables a coherent and holistic scheme of study which ensures that the students will find the coursework relevant.

Using a database management package, with command files, is also discussed from the point of view of using this as an introduction to learning programming. The fundamentals of a top-down learning environment with emphasis on files, both sequential and ISAM structured, is discussed. The student will be dealing with programming concepts at a much higher level than would be encountered in the initial stages of a commonly taught course in, for example, BASIC.

The use of a database management package is found to enable the computing course to focus on the design aspects of computer systems, rather than the current course emphasis on the bottom level programming details using languages such as BASIC. The design of processing as well as the design and implementation of systems can be used to develop valuable skills in the students.

The thesis concludes by examining how a teaching strategy centred around a database managment package could be used with the new GCSE Computer Studies course, which for the first time enables students to achieve the standard educational levels of attainment in computing without having to learn about programming in detail. A database management package is found to be an ideal basis for the GCSE course.

# TABLE OF CONTENTS

## LIST OF FIGURES

## Appendices

# LIST OF FIGURES

Dissertation for an M.Phil. research degree.

# 1. Chapter One - Introduction

## 1.1. Preamble

There has been much criticism of computing education, especially with regard to schools. Articles have appeared with titles like 'BASIC damages the Brain' (1.1. Atherton R 1982) and the general assertion is that computing courses are at least feeble and at worst damaging. Much of this criticism has focussed on the teaching of programming but recently developed courses tend to avoid teaching programming. They are still prone to criticism. The example can be cited of many universities who prefer their students not to have studied any form of computing course (and especially 'O' and 'A' level courses). Clearly there is a problem. There have been several well documented attempts to solve the problem, the proposed use of COMAL being the best known. These have tended to fail because teachers and lecturers are unlikely to move in any direction at the forefront of a dynamic subject area. BASIC is tried and tested and is preferred even if it fails the test.The programming language is only a small part of the overall problem, and one solution is to construct courses with no programming content at all. This is valid for courses where skills in programming do not need to be learnt. However, other courses require an appreciation of what programming is about and experience is desirable. In the case of 'O' level and 'A' level courses the syllabus dictates that a course without programming content is a non starter. What is required in order to bring about a fundamental change for the better is a tried and tested (successful) alternative. Surprisingly there is such an alternative. It has been around for a long time (in computing terms). It is even considered to be the next logical step in developing system implementations. If this is so then why not in education ?

## 1.2. Aims Of The MPhil

Microcomputer database systems (of which dBASE II is the most widely known) provide powerful applications tools. Such database packages can be used to both develop simple applications quickly, and also to develop complex systems requiring 'command files' containing program statements written in the query, manipulation, and report languages. The use of database systems should provide a good basis for developing courses in computer education at various levels. Such courses should compare well with those commonly found (which tend to involve programming in BASIC to a large extent).

The main assertions of the dissertation are that a course based around a database package could span all of the objectives of a typical computing course, and that experience of a database package could form a valuable part of any computer related course. In addition, such a course would tend to be 'holistic' in nature, and encourage a structured approach to the teaching methodology, which is missing at present. Most computing courses adopt a bottom-up approach to the subject, where different sections of the course are compartmentalized and taught separately from each other. In a holistic approach, the methodology would be to give an overview and develop expertise and skills in each differing area in a way which will allow the student to relate different aspects of the course into a common learning area. The holistic approach is defined and detailed in chapter five.

The main aim of the MPhil will be to investigate whether such a course of study could be developed, to consider teaching strategies for databased courses, and also to investigate how successful database packages have been when used in this way, with both experienced and novice users.

## 1.3. Objectives of the MPhil

Two main objectives can be established :

### 1.3.1. Literature Objective

A literature review - involving database fundamentals and also from the point of view of using database systems for computer education and introducing computer applications to new users.

### 1.3.2. The Structured Tools

A structured approach to teaching computing could be based around a varied set of tools. The thesis will discuss these tools :

A DBMS (dBASEII in particular)
Studying the use / success of dBASE II as a core part of a computing course, together with a survey of the opinions of commercial users to students taught in this way as compared particularly to BASIC.

The dBASE II Educational CAL package, studying the effectiveness of this 'on-line' teaching package.

Case Studies, with reference to their use with dBASE II or other database packages.

Programming, from the design point of view.

## 1.4. Thesis Overview.

Most educational users who have come across microcomputer relational database management systems (MicroDBMS) have realized the potential for using them in administrative functions. It is becoming clear that MicroDBMS have great potential at every stage in computing courses.

Students who have been exposed to either small or large doses have learned better and faster than would otherwise have been the case. This applies to students on literacy and on specialist computing courses.

Most students who meet computers are still being introduced through the use of BASIC. Students are generally taken through a detail orientated, bottom up learning sequence, and, if not baffled or confused by exposure to BASIC, they still fail to place the computer in its overall context.

On specialist computing courses students are generally taught programming from a bottom up approach, and are then expected to adopt a top down methodology to problem solving with the computer.

It is often the case that students learning BASIC meet files as the latter content of their course (if at all), although files are likely to be the most important aspect of any BASIC course. The introduction of LOGO and Prolog may change the situation, but these languages are taking a long time to establish themselves and teachers are seemingly reluctant to migrate from an 'industry recognized' language to an 'educational' language. MicroDBMS offer another and promising alternative solution.

Database is a word often abused, and MicroDBMS can be separated from the filing systems which are often referred to as databases. The key element is that of management of the database, and filing packages do not offer this comprehensive function.

Relational databases are the easiest to understand and apply from the novice's point of view. Of the MicroDBMS around, dBASEII has been marketed most successfully and is available cheaply on all educational CP/M based machines. While not at all perfect from the educational point of view, it has widespread industrial recognition.

Novice users should spend a lot of time with the common business

microcomputer packages: spreadsheet, word processor, filing system and graphics. This may be sufficient in the time available for the student's courses.

With careful planning, a MicroDBMS (such as dBASEII) can be introduced early in the course: (dBASEII has an associated computer aided learning package, dBASE Educational). This early introduction will have some dramatic effects.

Users are constrained to look at how to apply the technology to the problem, using a top down methodology. Attention is focused not just on the technology but on how to apply it.

The emphasis will be on deriving a solution, and implementation will be straightforward so long as the problem isn't too complex. The student is not bothered with the detail of how to get the job done at the early stage of the course, but is constrained to adopt a top down methodology right from the start in order to use the system to solve problems.

The students are introduced to files (both sequential and ISAM) straight away, and begin to manipulate files easily and with confidence at the early stages of the course.

The students can use the system to do useful and realistic tasks early on without getting bogged down in program detail.

Students on specialist computing courses who will be going on the do programming courses have already gained from manipulating files. They can also go on using the MicroDBMS to develop command files, although these should be kept simple, and all of the important structural concepts, for example:

| | | |
|---|---|---|
| IF | DO WHILE | DO CASE |
| ELSE | | |
| ENDIF | ENDDO | ENDCASE |

can be demonstrated and used quickly and easily - without having travelled through the detailed aspects of a programming language. How many students following a BASIC course really understand the concept of the IF ELSE ENDIF structure when they cannot implement it properly in their version of BASIC ? And the CASE structure ?

Students can be formed into teams to complete data processing tasks using the MicroDBMS and will learn how to work within a team and the importance of handling large amounts of data using a computer. How many students on conventional courses ever use more than a minimal amount of data to test their BASIC programs ? How many really learn about the problems of handling large amounts of data using a computer?

Potential employers (who are only too familiar with the 'Mickey Mouse' type of packages that students are usually subjected to) see a useful skill developed by the student.

There are other and more subtle benefits from using a MicroDBMS at the early stages of a computing course. which can only be appreciated after its use has been experienced by the teacher. Experience has shown that students using a MicroDBMS at an early stage of their course learn better, faster, and achieve a much higher all round knowledge of the technology. Students who then go on to BASIC or COBOL courses do better (they are already familiar with file manipulation) and want to adopt a top down methodology for program design. MicroDBMS seem to offer the opportunity of adopting a holistic approach to computing courses, which is missing at the present.

This overview was published in COMPUTING Magazine (1.2. Hitchman S 1985)

The next chapter reviews the current literature and discusses some of the course syllabi currently used.

## 2. Chapter Two : The State Of Computer Education

A quick look at recent articles in Computer Education (a magazine with wide educational circulation among schools and colleges) will show an overall picture of complete diversity in the educational useage of computers, and little conformity about what should actually be taught in schools. There has always been a strong tradition in England (as compared with countries where there is a strong centralized control over syllabi) of freedom within the classroom for the teacher to teach as they themselves see fit. Although groups may be taught to a provided syllabus, the subject matter and approach are those of the teacher. Providing the subject is relatively stable, it seems reasonably certain that different groups will receive similar teaching, perhaps with different emphasis. Within the realm of computing this freedom has proved a considerable drawback.

Two factors make this particularly so, firstly the lack of expertise among the teachers themselves (many of whom are ex teachers of another subject area) and secondly the very dynamic and rapidly changing nature of the subject itself. There are many syllabi which focus extraordinary attention on punch cards and paper tape as backing storage media because they have not been updated for several years.

A third and rather invidious factor has been the emphasis on teaching programming using BASIC. The all purpose nature of the language could be thought of as a plus, but in reality this lets loose the classroom teacher to delve into every and any corner of the language. Indeed the scope of the language provides course material for many more years than the run of most computing courses.

### 2.1. Introduction

The main aim of the thesis is to show that by basing a computing course on a database package, and remembering that the present alternative is to use BASIC, then the quality of the student's

experience will improve. Further, for courses that are presently based on integrated microcomputer packages, then a database management system should form the main core of student attention.

In this chapter an account of common syllabi, and current educational trends is given. In summary the feasibility of using a database management system is discussed.

## 2.2. Example Curriculum Documents

The main features of the syllabii are discussed here. Those chosen are a sample from many.

### 2.2.1. CS1

This information, from an interesting article (2.1. Not Attrib,1984) concerns an American University syllabus in Computing Studies called CS1. This is a first course (lasting one semester) in Computer Science. The article concerns a proposed new syllabus and so can be regarded as the most recent thinking. The article does not outline the contents of the syllabus, but is interesting in that it specifies the exact mathematical requirements for any student entering the course. They are as follows:

Elementary

- summation
- subscripts
- simple functions (ABS INT)
- logarithms (eg. for sorts)
- prime numbers
- greatest mean divisor (for the euclidian algorithm)
- the floor and ceiling function

General

- functions (correspondence, and mapping)
- sets (data types)

Algebra
- matrix
- polish notation
- congruence (random number generators)

Summation and Limits
- elementary summation calculus
- f(a)
- harmonic numbers (for quicksort)

Numbers and Number Systems
- positional notation
- bases

Logic and Boolean Algebra
- logic for design
- basic logic

Probability
- test data

Combinations
- algorithms

Graph Theory
- basic concepts
- trees

Difference Equations and Recurrence Relations
- tower of hanoi, quicksort (for example)

This is interesting because it reflects the current thinking in many circles (especially at a higher academic level) that computing requires a high level of mathematics knowledge. Many school computing teachers are ex mathematicians. The list of requirements for that

University level course would restrict all students to a reasonably high level of mathematical skills and knowledge. Strangely this is not reflected in the commercial environment. Many (if not most) commercially successful computer people do not have such a high degree of mathematical skills. Indeed, it is often the case that the content and bias of such courses is frowned on by the business world because the students are not being taught the skills they need in order to apply their knowledge to real situations. In other words, there has been a tradition, which still persists, to regard computing as an offshoot of mathematics.

It is not the intention here to look at the conceptual basis of high level specialist computing courses. Rather this is important because it has affected the way in which schools and colleges have approached computing education. It has always been the case that courses at the lower end are formulated with the higher end in mind. Instead of relying on mathematical subject areas to provide students with the required computing course inputs, these have been reflected in 'O' and 'A' level syllabii so that these have been very mathematically based. It is very questionable whether, for example, and 'O' level student needs to understand the 'bubble sort'. Commercial computer users utilize either ISAM files or a utility sort. Such an element (regarded as essential by many teachers) has arrived there because of the mathematical background behind the development of the subject.

It is against this background that programming has become such an important part of many computing courses.

This is really the starting point for the discussion, the following syllabii have all evolved from this mathematically inclined viewpoint.

In a second article in the same series (2.2. Koffman et al. 1984) the objectives of the course are outlined. The course is designed to emphasize programming methodologies and problem solving, it is not designed to be a computer literacy course or a service course for other disciplines. It incorporates the current trend towards software

engineering,data abstraction, and good programming style.
The objectives are :

To introduce a disciplined approach to problem solving methods and algorithm development.

To introduce procedural data abstraction

To teach program design, coding, debugging, testing and documentation using good programming style

To teach a block structured high level programming language

To provide a familiarity with the evolution of computer hard and software

To provide a foundation for further studies in computer science.

**Fig. 2-1: CS1 Syllabus Objectives**

The implication here is that all computer scientists need a very extensive grasp of mathematics. Among the objectives is one that states that students should be able to formulate, represent, and solve problems using the computer. Top down design and stepwise refinement will be stressed. The programming language used should not be BASIC or FORTRAN, but should be similar to either PASCAL, PL/1 or ADA. Strangely, the prerequisite programming course is in BASIC.

### 2.2.2. Accounting

Before looking at broad based computing courses, it is worth looking at a specialist course which contains some computing. The course is aimed at accountants, and is briefly outlined in Computer Education (2.3. Steedle L F et al 1984)

The article states that because of the dramatic evolution of the computer and its rapid expansion into traditional accounting areas it

is crucial that all accounting programs meet reasonable educational objectives in the computer area.

The article is based around the report of a professional American accounting body (the American Accounting Association), and goes on to list three broad objectives:

computer orientation
computer programming
information systems

Although no detail is given the inclusion of programming seems extraordinary since accountants will be system users.

### 2.2.3. Computer Literacy

Computer Literacy is currently considered to be a 'subject' in its own right. Most schools and colleges now define and teach the subject. It is my own view that literacy should be achieved by the use of computer systems within other syllabus areas, rather than through a separate course of study. The Further Education Unit (FEU) has recently developed and circulated a computer literacy syllabus, primarily aimed at students taking YTS courses, although designed to be suited to any initial literacy level. This is a lengthy document, aligned with a piece of hardware (the BBC Model B) and is worth considering in some detail. This is because the FEU have defined and categorised the subject of 'computer literacy' and have established an approach to teaching it. This approach is widely used, and an understanding of the FEU syllabus will lead to an understanding of much of what is taught in schools and colleges. Much of what follows is taken from the FEU discussion document (2.4. F.E.U 1984). The section on programming is particularly interesting.

### 2.2.4. FEU Definition of Computer Literacy

Computer Literacy has been comprehensively defined in the document, as a term which evolved during the 1980's with the advent of the BBC Computer Literacy Project. It is defined in these terms :

"Understanding what computer systems are"

The FEU state that "We need to know something about how computers work and the vocabulary associated with them before we can use them." This is a debatable point, and many would consider that the user should be shielded from such things as much as possible, and that the system should be as 'transparent' as possible. The knowledge required will ".... include an understanding of the functions and relationships between the main parts of a computer system".

"Usage of computer vocabulary"

"Operating a computer in a work related situation"

"The learning of operational skills should take place in a work related situation (or at least in the context of using the computer to perform a useful task). Examples of such applications are the use of simple word processing, spreadsheet and database packages." In this context database package refers to a filing package.

"Appreciating what a program is and why it works"

It is noticable that although the term 'appreciation' is used here, The FEU advise that programming skills be taught. "It should be stressed that the inclusion of the appreciation of programming means a minimal experience and that the implementation can be achieved in a variety of ways. It does not necessarily mean the writing of programs in BASIC." Although this general outline suggests one direction, the FEU implementation takes a quite different direction in that BASIC programming skills form a major part of the syllabus. "The rationale for including programming is that to fully understand and to confidently use pre-written software packages demands an appreciation

of how that package has been produced and its potential vulnerability. A demistifying of what would otherwise be a black box has proved successful and has gained almost unanimous support in our field testing and evaluation." Most implementors may have had a vested interest in a continuing programming skills content of such courses, no sources are quoted and other experience points to the opposite effect, ie. students become very confused when dealing with BASIC programming at this level.

"At the same time the recent availability of the languages PROLOG and LOGO on microcomputers means that programming can be introduced in rather more painless and interesting ways as an alternative to BASIC." This is a particularly interesting point with reference to the chapter on programming languages and the command file capability of dBASE II. "PROLOG and LOGO are languages which are based on mathematical logic and are nearer to everyday 'human' expressions of problems whereas BASIC although the universal language of micros is more influenced in its structure by the hardware." This again highlights the in built assumption of the need for a 'mathematical' basis for an understanding of computing, and now at this much lower level.

> "Being aware of applications of computing in commerce, industry and other settings."

> "Being aware of current trends in information technology and its social implications."

"It is in these latter two aims that training and education cannot be divorced if students are not to loose sight of the context in which they are working."

### 2.2.5. Implication of Computer Literacy Courses

The aims outlined above, by the FEU are not only supported by stand alone literacy courses. They can be found as the core parts in most computing courses. There is generally an assumption that programming

- that is the teaching of programming skills, is important and mathematically based.

The fact that the Range of Uses aim (which gives the framework in which to use computers) comes fourth, whilst the detailed knowledge of such things as hardware and programming come first, suggests the bottom-up approach taken by so many computing courses. Of course, the implementation does not have to be in the same order as the aims, but I would have expected the framework of what computers are used for to have been placed firmly first and to have set the context for the remainder of the literacy course.

There is a lack of real emphasis on using computers to do things, and provide skills in expertise, and too much emphasis on learning about the lower level tools (like programming) which many of the students will never experience. It is rather like teaching about compression ratios, carborator mixtures, and brake horsepower factors, when what the student wants to know is how to drive the car.

A further insight into the thinking behind the literacy scheme is given by the FEU discussion on why literacy is necessary. This is detailed in the following section.

### 2.2.5.1. Why Computer Literacy ?

"It is a first essential step in the use of Computing as a tool across the curricula, and the FEU recommend it as part of the common core for all vocational preparation schemes. All of the new vocational schemes:

YTS - Youth Training Scheme
CPVE - The Certificate Of Pre-Vocational Education
TVEI - Technical and Vocational Education Initiative for 14-18 year olds

have this computer literacy as a part of the common core."

"Evaluations by FEU have shown that computer literacy has enabled students to cope confidently with the technology when they have encountered it in its various forms at work and at home, and to adopt a considered view of its social implications. It has enhanced their employment potential, particularly in small business where the need is for the possession of a wide range of microcomputing and other skills. It has opened the door for vocational preparation students with identified potential in IT to pursue further FE qualifications and hence obtain higher quality jobs (which begs the question of what went wrong in school ?)." The FEU conclude by saying "It is heartening that it is through computer literacy that an otherwise hidden potential is identified....adults quickly become computer literate and therefore confident in relating to the computer at all levels."

"Various local initiatives (by centres offering the three mentioned schemes) have been made along similar lines to those outlined."

The FEU lines of though here are quite clear, and reflect the concept that computer literacy is a subject in its own right, and that essentially the subject content is a 'scaled down' version of the O or CSE syllabi. Significantly it is not defined on the criteria of what the user needs to know in order to make the machine do what the user wants. Literacy is defined as basic background 'fundamental' knowledge.

This is generally the case, but it is also true that computer literacy is provided when the student uses the technology to do something within any (other) subject area. In other words, the growing use of CAL packages within all subject areas will produce, through exposure, the computer literate student. This is in contrast to the FEU formula, which regards 'CAL - Spinoff' as the reverse process. The term 'CAL - Spinoff' is the FEU term for the advantages to be gained by the student from the literacy course when they meet CAL in other subject areas.

### 2.2.5.2. Literacy Implementation

In order to implement their scheme the FEU have developed a module teaching scheme, which contains around 60 modules. Together they cover all of the course objectives. BASIC programs have been developed by the FEU to run on a BBC B machine and are used throughout the modules. (It is worth noting that I was unable to obtain a working copy of the program suite which was issued on tape despite several attempts. At least one lecturer was unable to implement their literacy course because the programs could not be obtained. In addition to the written tapes, several packages are recommended, including:

WORDWISE wordprocessor - text formatter (unlikely to be commercially compatible)

BEEBCALC spreadsheet - a very skeletal system

VU-FILE 'Database' - also low level

Staff development is seen as a definite need, and a strategy for this is also outlined by FEU.

### 2.2.5.3. The FEU Modules

Because it is important to get an overall view of the implementation of the FEU course, the modules are now discussed. The modules are split into nine sections which comprise:

| | |
|---|---|
| Introduction | 2 |
| Computer Basics | 8 |
| Understanding Programs | 12 |
| Business Applications | 8 |
| Office Applications | 4 |
| Industrial Applications | 6 |
| Other Applications | 7 |
| Social Implications | 3 |
| Visits | 4 |

The numbers indicate the number of modules attached to each section. Understanding Programs shows its true significance by the number of modules attached to it - more than any other section. Indeed, the implementation indicates a divergence from the literacy aims. Packages, on the other hand, seem to be under represented.

The programming section modules are :

What is a program
Drawing a flowchart
Program control
Writing a program
Further flowcharting
Further programming
Introduction to graphics programming
Writing a graphics program
Writing a project program
Testing a project program
Programming languages
An alternative language

There is no mention of designing a program (flowcharting is a poor method of designing BASIC programs). The question can be asked - why write a graphics program at all (at this level), why not use a graphics package and why is so much of the course time spent on program writing ?

### 2.2.5.4. FEU Courses - Summary

The FEU have produced a complex and detailed package for teaching Computer Literacy, aimed at the three main vocational training areas, but also suitable for any Literacy course. The modules are well produced, but I would argue that the scheme fails for various reasons:

Firstly the reliance on the BBC hardware. Clearly this is not a bad idea in principle since most of the sites where the module will be

taught have this equipment. However, inspection of the modules in detail reveals that the applications outlined (especially the supplied BASIC programs) are simplistic, and do not reflect real business situations. The BBC packages are not very good, the WORDWISE wordprocessor, for example, is nothing like the typical commercial counterpart.

Secondly, the failure to mention the words Application, or Application Package. All of the vocational course users, with a very few exceptions, will use packages. They should form a much larger part of the course as a whole. It is notable that the FEU concentrate some attention on those on the course who have continued into higher levels of Computer Training - this can only be a small portion of the total student throughput.

Thirdly, and whatever FEU would have us believe, the inclusion of all that BASIC is a disaster. Most of the students on the courses I have seen in operation get totally baffled by the programming aspect (indeed, many of their course lecturer's lack expertise too). A brief glance at the modules in question leads to the conclusion that there is indeed a great emphasis on this aspect of the course. There are many reasons for this. The course designers are probably all programmers and they like programming. Many of those used in pilot testing were likewise programmers (in BASIC). Throughout the educational computing environment there is a vested interest to maintain programming as a major part of any computing course. It is rather like teaching someone to drive a car by teaching them how to put the engine together.

Finally, it is worth saying that, apart from the programming aspect, the use of WORDSTAR, dBASE II, and perhaps a spreadsheet, would more than cover all of the aspects required by the literacy course. This is apparent from the objectives and modules. Indeed, had dBASE II been used from the outset a much more comprehensive (and working !) set of accompanying software would now have been available. As far as the programming is concerned, the modules are pitched at such a low

level that all of the section objectives could be met by using the command language of dBASE II, with the exception of the graphics part. Of course, dBASEII is not available on the BBC hardware.

### 2.2.6. BTEC Courses

The main institutions dealing with full time courses in Colleges used to be BEC (Business Education Council) and TEC (Technician Education Council). Computing, as a subject area, was deemed to fall between the two, and so B/TEC (Business and Technician Education Council) created a joint committee to oversee the syllabus and manage the college courses. BTEC (without the /) has now taken over from BEC and TEC through amalgamation, although it will be some years before the end users notice the difference. The BTEC courses are interesting, because although they are based on a centralized syllabus, the individual colleges write their own internal assessments (in course assignments and examinations). Although this assessment work is moderated by an external BTEC moderator, in practice the emphasis of a course can vary between colleges, and indeed revolve around the lecturers in the college. There is, therefore, scope to make these courses responsive to the local needs of business and commerce.

#### 2.2.6.1. BTEC Course Outline

The BTEC course in Computing and information systems is a specialist Computing course (2.5. BTEC 1981), designed to produce trainee operators and trainee programmers. This is a brief outline of the course. In the first year of the two year full-time diploma the students take these modules:

Introduction To Computing (double module)
People and Communications
Information in Organization
Quantitative Methods
Programming Concepts (half module)

In the second year:

Programming Concepts (second half module)
COBOL Concepts
Programming Project
* Small Business Systems Concepts (double module)
* Small Business Systems Project (work experience)
* Option Module (microelectronics)

Each college offering the course may vary the *'d modules, but the other modules are generally considered as core modules. The syllabus is outdated but due for replacement in 1987. One option module offered by BTEC is called Computer Operations, and instructs that students obtain detailed work experience of using paper tape readers and punch card readers, because when the syllabus was written they were the latest technology !

This dated syllabus does not constrain the college too much, and they may legally vary the content of the modules by some 50% from those stated. The quality and emphasis of the courses varies a lot from college to college. This course will be dealt with in more detail in chapter four.

**2.2.6.2. BEC Course Outline**

Although BEC is now BTEC as well (!), this is a useful title to distinguish another Computing course. This is a service course, designed for students taking the BTEC Diploma in Business Studies. The course is titled IP1 (Information Processing 1) and is the first year module which replaced an older Data Processing module in 1982. It is one of the most recent BTEC courses available.

One of the major changes from the older course was that programming was removed from the syllabus, and was generally replaced by a much more detailed look at the three integrated microcomputer packages. This reflects the view of most academic staff involved with business

studies courses that their students do not need to know how to program. This is an interesting contrast to other lines of academic thought (particularly the accountant syllabus and that produced by the FEU). It is certainly a reflection of the fact that the advisors to BTEC, who helped to define the syllabus and who come from the world of business, did not see the necessity for the inclusion of programming in the course. Indeed college experience of the older Data Processing course is generally that the programming aspect was poorly appreciated by the students despite, or perhaps because of, being taught by skilled programmers.

### 2.3. New RSA Courses

The RSA (Royal Society Of Arts) have introduced their own Computer Literacy and Information Technology Scheme, aimed at students who would have completed, for example, the RSA typing courses. The literacy scheme is not examined here, since this has been dealt with in the section previously dealing with the FEU initiative. It is worth noting that the literacy course, designed as a short course (unlike the FEU scheme) involves no programming, but concentrates on keyboard skills, microcomputer operations skills, and common microcomputer packages. This results in the RSA Literacy course being amoung the few courses designed in this way, and reflects the very latest thinking on literacy courses.

The two RSA courses considered in detail here (2.6. RSA 1985), are designed for in-service education for teachers and are split into Computer Science and Use Of Computers.

The Computer Science course is based around the traditional schema for an 'A' Level Examination (the course is designed to equip teachers of such courses). There is an emphasis on structured programming. The Use Of Computers syllabus is rather similar to the IP1 course outlined above, although there is more emphasis on packages experience, and the inclusion of study on educational package use and assessment.

### 2.3.1. Teachers taught the RSA way

A report in Computing (2.7. Mill J 1985) outlines the RSA course concepts in the light of the "Micros in Schools" scheme. The government scheme placed computers in school, but provided no training in use for the teachers. The RSA scheme was developed to meet this need. The modular structure of the courses mean that any student teacher can follow a course tailored to their specific needs. The current teacher experience is that training given has been "...out of date...at a fairly basic level..and you don't have time to get a wider perspective on the use of computers in the real world." Jenny Mill concludes her article by stating that "..the lack of funding available for in-service training and for teachers to be given time off to attend courses means that the potential of micros in schools is far from being fully exploited."

Jenny Mills describes the effect of the courses, the next section looks at the course content.

### 2.3.2. RSA Course Content (Teachers Courses)

The Teacher's Diploma in Computer Studies course consists of several parts, each with an accompanying target of hours to be used in study (shown in brackets).

Information Systems and Computer Applications (40)
(Systems Analysis including Databases)

Computer Architecture and Systems Software (30)

Programming and Problem Analysis (30)

Computers in Society (20)

Methodology (30)
(Dealing with the methodology of teaching A level)

The details of the parts of the course are not really spelt out, and the lecturer has a great deal of freedom in their approach. However, the first two parts (on theory) are examined by RSA papers.

The Use of Computers course is aimed at testing the ability of candidates to evaluate and implement the application of the computer to various disciplines, manage computing resources and assess the potential impact of educational computing in curriculum development and teaching methods.

The course is divided into 4 parts :

Computers and Learning (30)
This includes the use of databases as sources of information

The computer as an aid to teachers (40)

Managing computer resources in an educational institute (30)

Evaluation, design and production of educational packages. (50)
This includes the use of authoring packages, and does not include the production of package software.

### 2.3.3. RSA Summary

The RSA have recently piloted a completely modular certificate / diploma course which gives much greater flexibility to the teacher trainers. It is still the case, however, that teachers employers are not prepared to pay the price for training, and rely on the goodwill of teachers to attend these courses in their own time.

The RSA syllabus course documents are included as appendix three in

the thesis. There are two separate documents, one each for the literacy course, and the teachers course.

The RSA courses are the latest indicator in the way in which computing is being approached as a subject in schools. The emphasis is towards packages and using the computer as a tool but there is no conceptual foundation to the content of the syllabi, as there is in the FEU documents. The lecturer of the RSA courses is given no guidance on how to approach the course content, or where the stress needs to be put. Jenny Mills indicates that the promise of the syllabus is not being carried to the teachers, if only because of the short amount of time available to the student teachers on the course.

## 2.4. Current Trends In Computing Education

After that brief look at some of the currently taught syllabii, it is now the time to look at the literature for guidance as to the current trends in educational computing. Three trends are immediately apparent, they are a move towards using 'databases' for teaching in all subject areas, a move to using structured programming techniques, and a move towards designing courses with no programming content for use by non specialists (not universal).

One other thing is apparent, and that is the lack of any real direction in the subject. Almost every article reflects the bottom up approach that seems to prevail, and the articles deal with the technical detail of, for example, how to fit a light pen to a BBC or a program for statistical analysis.

### 2.4.1. The COMAL Experience

There are some notable exceptions. First attempts to change the way that Computer Education was taught came to light around 1980 with the movement towards using structured programming techniques, and the realization that something was wrong with BASIC in schools.In 1982 Roy Atherton (2.8. Atherton R 1982) published an article titled 'BASIC

damages the brain' (there had been a previously titled article in a commercial magazine).

The theme of the article was simply that students with long, or even short, exposure to BASIC would be so damaged by the experience that they may never become 'good' programmers, or even understand what programming was all about. A glance at any student program of the time would support this. Further evidence began to emerge as teachers in schools produced educational software. With a few exceptions the number of program lines was only marginally greater than the number of GOTO statements, few of the programs even worked or were documented, and none were maintainable. It was quite clear that the skills and knowledge obtained by students taking BASIC courses was detrimental in later studies.

Roy Atherton realized that little would change until the language changed (even though structured programming can be taught with BASIC) and he has been proved right. At the time there was a viable European alternative which attracted his attention, just before the Government (and Local Education Authorities) made decisions on what hardware (but not what software) to buy. A particularly articulate and clear thinking Danish teacher, Borge Christensen, had begun to define an educational language as early as 1974. This process began when he made amendments to an interpreter (by patching on extensions) and ended with a completely new language. Christensen aimed to take the best of BASIC's simplicity (and programming environment) together with notions of good programming structure and good programming practice as represented by PASCAL (2.9. Peltu M 1980).

He soon produced the first version of the language (with the help of Benedict Loftstedt) which he called COMAL (COMmon Algorithmic Language). This was an extension of DGXBASIC on an RC(NOVA) computer and was working in 1974. It became very popular in Danish schools (as have some extended versions of BASIC in Britain). For example, unlike the BASIC structure of:

linenumber IF condition THEN statements ELSE statements

He implemented the structure

```
IF condition

    statement
    statement

ELSE

    statement
    statement

ENDIF
```

Surprisingly this is not at all difficult to do (a lecturer at Basingstoke College has recently done the same to TANDY BASIC) although having said that I wouldn't like to do it myself. In other words any computer manufacturer could have implemented it and other structured features in their version of BASIC. COMAL has since been improved and standardized and has gained popularity amongst many educationalist in Europe, because it enables good programming techniques to be taught using a language which has the directness and familiarity of BASIC.

In an article (2.10. Christensen B R 1982) concerned with BBC BASIC, Christensen made an impassioned plea for the adoption of a structured language. Outlining the success in Denmark of COMAL, Christensen went on the explain that Europe has actually been consistently more successful in designing languages (remember PASCAL) than America, and it was unwise to cling to the American BASIC when the European COMAL was so much better.

Concerning BBC BASIC and the BBC Literacy project he said "(we should be) .. fully aware of the crucial importance of the programming language that may be used by hundreds of thousands of viewers. Not

only for programming purposes but also for the communication of ideas".

Roy Atherton had said some years previously that the Government, and even a LEA was in a position to demand any kind of software implementation they liked. NOVA marketed a microcomputer with COMAL which was years ahead (and still is) of any of the British machines then available and at the time was cheaper. Even ICL were marketing a COMAL based micro in Europe that could have been marketed in Britain.

So, what went wrong ? There is no doubt in my mind that COMAL - at the time - was a major step forward. The Programming environment offered by the COMAL machines was superior to anything available even today. Lots of factors affected the lack of uptake of COMAL in Britain.

Firstly, Universities didn't offer useful guidance. Secondly, the people on committees in LEA's and in Government were constrained (for example the hardware must be British). There was an over emphasis on the hardware without attention to the software. Thirdly, the attention of many educationalists was on turning non specialists away from programming at all. Fourthy, people were reluctant to adopt a new - an educational language - which was not known or experienced by industry. It was a time when educationalists were afraid to make a mistake in a field they did not fully understand. Finally, and most importantly, the marketing was wrong. COMAL was marketed as an enhanced BASIC. If BASIC is discredited then so is an enhancement. If COMAL had been marketed as a beginners PASCAL then it may have succeeded. If ICL had marketed their COMAL micro in Britain in time for the Government support decision (for a computer in every school) then COMAL might have succeeded.

It is a sad British story which has now been superceeded with the advent of newly available (to schools) languages like PROLOG and LOGO, but it is a reflection of the state of computer educationalists in

Britain that they could not move themselves to adopt the simple (though brilliant) ideas of Borge Christensen, and instead were dictated to by the manufacturers of hardware systems.

### 2.4.2. Other Contributions

There is no real evidence of any major trends away from the current status of computing education. In the Computing Newsletter (2.11. Not Attrib 1984.) there is a brief article concerned with the concepts that students using spreadsheet packages should not just use them for practical 'hands-on' experience of computing, but that they should form an integral part of the teaching of concepts on the course.

In another article in the same newsletter, but from October (2.12. Not Attrib 1983) there is a report of a Dr.F.MCFadden,who is teaching an American database course (part of a degree module) who has been using dBASE II with some success in demonstrating the concepts he has been teaching concerning databases. Dr.McFadden had previously only used lectures to teach his students about databases.

In an article entitled 'DBMS For Kids' (2.13.,Holmes E 1985) the growth of the use of what are really filing systems with data is noted in American schools. This is considered to be a growth software area together with more 'Kids' buying and using wordprocessor packages, rather than games. The filing packages are really 'encyclopedia' replacements. The trend here is for children to move away from using home computers to play games, and away from the more usual 'drill and practice' kind of educational program commonly found, and a movement towards more intelligent and interactive educational systems. Examples of software packages for the Apple 2E are quoted, and some similar ones have been recently advertised in Britain for the BBC using the VU-FILE package. For example, a database is supplied with details of all of he countries in the world, together with various statistical factors, and the student can search the database using various criteria, eg. find all of the countries with a population greater than 60 Million. Essentially crude (because they are based on

crude filing systems) but an interesting development.

### 2.4.3. The Female Aspect

A student from one College recently visited another college and showed great surprise (and enthusiasm) because there was a female student on the Computing course. There are few female students pursuing college courses in Computing . Recent research by Lorraine Culley (2.14. Culley L 1985) highlights the situation and gives several pointers to what actually happens in school computer education.

Her research at eight secondary schools in England points to the usual problems of career guidance and sexism in careers literature, parental influence and the image of computing as a scientific and/or mathematical pursuit. However, she goes on to distinguish a more pervasive force against females in the Computer Classroom with a systematic difference in the experience of boys and girls in the Computing Lesson.

This covers all groups, specialist examination classes as well as 'IT' classes.

Girls were losing out when there was 'competition' for scarce computing resources, perhaps because it is unladylike to rush to the terminals. Boys were laying claim to the best or the most of the available machines. Girls tended to form single sex teams, and if in a team with boys took a back seat in computing use. Whilst girls coped well with sharing terminals, boys tended to squabble a lot, and much of the teacher's attention was given over to sorting out male disputes at the terminal. Indeed in one observed lesson the teacher spent so much time sorting out the boys that two group of girls were left for the entire lesson without being able to load a program from tape and the teacher did not notice.

During class discussion the girls made little contribution, the boys dominating the discussion. Girls would linger and show little

enthusiasm for the lesson, whereas boys would be eager to begin.

It seems that there is more here than in other subject areas. The boys often treat the computer experience as a competitive act, rather like school sports, in which peer competition is the main determinate of what happens. It is in this kind of atmosphere that programming concepts are taught. The atmosphere is not condusive to the carefully designed and considered implementation of program specifications. I think that the research points to an underlying flaw in the way in which computing is being taught in schools.

### 2.4.4. The Holistic Approach

In one of the very few articles concerned with computing education in broad terms, H.W.Lawson outlines his 'Holistic' approach to teaching computing - to engineering students. (2.15 Lawson H W 1985)

The article is aimed at teaching engineering students about computing systems, but they are subjected to wordprocessing and other Computer Literacy areas. Lawson states 'One of the basic problems with computer education at all levels (including professionals-to-be) is that the process has been largely detail oriented with little attention given to concepts and principles. Our approach has often followed the stratification of the various separate specialities of the professionals involved from basic levels of circuitry to higher levels of application and utilisation. Try to imagine the 'perfect' computer professional who knows everything about all aspects of computing. Each component of knowledge has some conventional educational starting point. The problem with beginning with these detailed subjects is that it takes a long time before they are mastered and that during this detailed education, the student has difficulty in relating this specific knowledge to the big picture.'

This echoes the concept that by teaching programming as a major or perhaps first part of a computing course the student will not relate what they are learning to the big picture - that the program is

supposed to do something useful for the user, be easily maintained, be developed as part of a team project, and so on.

Lawson goes on to say that 'A far better approach would be to present a holistic core consisting of the essential ideas of each subject so that the big picture is clear from the start. This core, being a good starting point for professionals to be, would also serve as a core for providing basic computer literacy. The real problem is in finding an approach to presenting the core that can be mastered in reasonable amount of time by all students according to the following truism :

> PEOPLE LEARN BEST WHEN NEW CONCEPTS ARE PRESENTED IN TERMS OF WHAT THEY ALREADY KNOW

Several years of experimentation to find an appropriate approach has resulted in a process and system-oriented approach ..'

Lawson then goes on the detail the core scheme, which is actually geared towards an understanding of the hardware, for example mentioning parallel processing, simplex and asynchronous control almost straight away. The content may not be appropriate for students in other disciplines but the concept certainly is and that is where we can begin to look at how a database management system will fit into the computing education framework, not as just one more package, but as a major element in the scheme of things.

## 2.5. Experts Appraisal

In a recent article in COMPUTING (2.16 European Communities Commission 1985) the current thinking of a European educationalists seminar is detailed. Computers were thought to be important for children at the earliest primary stages. Delegates agreed that the most successful programs were those which put the children in charge of their own learning, and allowed them to explore concepts and ideas. The common packages, such as databases, were considered important.

Children should learn where to find information, assess its value and use it to solve problems. It was important not to confuse programming skills with information and computer skills.

Database packages, used presumably in a similar way to Quest by the river, clearly have an important part to play in the view of these educationalists.

## 2.6. Literacy From Other Subject Areas

Diana Freeman and John Levett form the Advisory Unit for Computer Based Education (AUCBE) (2.17. Freeman D & Levett J 1985) consider that the availability of information handling on computer can give children the power to seek, organize, question and draw conclusions from data. These are important skills and tools for knowledge in subject areas like geography and history. They go on to say that the computer are also common to all disciplines. It is apparent that information handling on the computer is a common core within the curriculum and is subject-independent". This has important implications for the computer subject specialist. The students who use computers in this way in other disciplines are learning much of what has been taught within the computer subject specialist area.

With regard to learning the command language of a database package, they say that "It would appear that in order for children to employ such syntax for their own purposes the conditions under which it is encountered are crucial. Thus it is important that the child can relate the command to the result that corresponds to a stated request of data (a mental picture) and that the child can form a substitution for the syntax (a metaphor)". In other words the use of a case study which the student can relate to is more important than learning a package for its own sake. Freeman and Tag (2.18. Freeman D and Tagg W 1985) say that "Databases, just as other teaching materials, have the most impact when the information is relevant to the pupils. Data that is immediately related to the work that takes place in the classroom, but which extends the use of existing materials, or data that the

children have collected themselves, comes into this category."

## 2.7. How a database fits in

The training needs for information technology are still unclear. Jenny Mill (2.19. Mill J 1986) questions the relevance for paper qualifications in relation to the recent need for stock market and allied systems. The experience of the potential employee in very specific areas was the major and often only consideration in job recruitment.

For all of the courses described, and indeed for all computer based courses, the starting point for the course and for the students could be an integrated microcomputer software package and a database management system. This would consist of something along the lines of Ashton Tate's FRAMEWORK, which includes Outliner, Word Processor, Spreadsheet, Graphics, and Filing packages, all of which integrate. The dBASE II or dBASE III package will also integrate into the FRAMEWORK system.

The two most important aspects of the software would be considered to be the wordprocessor (which would at least be used to write programs on, for the program inclined) which might form a starting point for the course, with hands on experience. At this stage the skills and knowledge would be related only to learning the package, and associated wordprocessing concepts. Following this the filing system, or data base management system can be introduced, in a way which stresses the links between the packages. This filing system would only be used if seen as a sub-set of the management system and as a simple introduction to the management package. Only after the student has expertise in these two packages, and perhaps after some discussion and formal teaching of database concepts, would the student go on to any other aspect of the course.

The aim of this would be to stress the 'holistic' (to borrow Mr.Lawson's phrase) approach to the computing course. The student

will be learning how to do things with the computer, as well as learning how to communicate and control the computing environment. All other aspects of the course, whatever they may be, and including programming, can be built around this integrated package view of the system.

## 2.8. Educational Implications

The educational implications of using this different approach, and stressing the requirements of a database approach will be reflected in many ways. Among the most important will be :

To focus the student's attention on using the computer to do something useful straight away.

To get the student to produce a 'system' (for example a memo 'mailshot' system) which works and is useful and soon after beginning the course.

To get the student to consider how to do it.

To get the student's to work as a team on a data processing project and so be working at the top level of the computing system, and working down towards the detail.

To wean the student away from the detail at the bottom level and towards the top level, since the database management system will deal with the detail for the student.

In this way a firm foundation of skill, expertise and understanding of where the computer fits in, will be obtained. It is on this foundation that the student can progress to more detailed parts of whatever course they are undertaking.

This would be in opposition to the kind of ideas formulated by, for example, the FEU, who have modularised the course and work from a

bottom up approach to an overall course whole.

The next chapter looks at the conceptual background that would be required of the teacher (and to a lesser extent the student) using the database approach.

## 3. Chapter Three : Changing Emphasis

This chapter looks at the implications of using a DBMS in an educational course. The conceptual framework of the course will change in emphasis and this is discussed. The next chapter looks at how this change in emphasis was used in teaching a particular group of students. It is assumed that the reader is familiar both with the content of a 'usual' school and college course (of which a flavour was given in the previous chapter) and with DBMS.

The way in which the computing teacher views a computing course should depend on the content of the course. For example, many school level computing courses still use the content of systems and programming flowcharts, and this affects the view of the teacher to the processes of designing programs and systems. If the ideas of the thesis were to be implemented then the content of the computing course would change. For example, in order to adopt the more modern analysis methods associated with database systems, entity relationship and data flow diagrams might be used. The way in which the teacher would view the course would also change.

The thesis is suggesting the introduction of newer techniques as a large part of the computing course, because a database system is used. This will mean that the teacher of the course will need to change the way in which the computing course is perceived, by the teacher and by the student. These changes will move the computing course towards newer techniques involved in analysis, such as entity relationships analysis.

In this chapter an outline of the required fundamental view of the computing course is given. The teacher of a current computing course will need to adopt these fundamental concepts much more closely than at present, and indeed many of these concepts (normalization for example) will not be commonly used by teachers in schools.

The concept of the Information System should be the core and starting

point for a computing course. It is important to make the point that a teacher using a package like dBASEII (or dBASEIII) will not be viewing the fundamental concepts correctly if they adopt the view of the package. For example, in dBASEIII there is some confusion surrounding the concepts of relationships and views.

Developing a computer based system has its associated methodologies and problems. Much has been written concerning the problems associated with the implementation of computer systems,here is a favourite example :

"A large part of the problem is that computers were born in an era of technical optimism. At that time man was solving more and more of his problems using technology, progressively harnessing its power. Computers in those optimistic days were considered as a bright new invention with enormous potential.. ..... The problem is perhaps that the computer was seen not so much as a tool for man, requiring human involvement, but as a replacement for god." (3.1. Crowe T & Davison D E 1980).

This point of view is reflected in the educational establishment where the computer has been seen not so much as a tool for learning but as an end in itself.

It is rather similar to the teacher of a class of motor mechanics who spend their entire course time studying and working on a car engine mounted on a bench in the classroom. At the end of their course none of the students is aware of the fact that the engine actually fits into a car and is designed to power it along the road (with the aid of the driver of course). None of the students understands the engine in the context of where it will be found and what it will do. Viewing data as a resource is analageous to viewing the car as a resource.

### 3.1. Data As A Resource

Every organization has a pool of resources that it must manage effectively to achieve its objectives. A student taking a computing course is in just such a position. All resources incur cost and are of value to the organization.

The value of data is unique in an organization since the entire organization depends on its availability for the management of other resources. Data and information can be distinguished, for example (3.2.,McFadden F R & Hoffer J A 1985):

> Data are facts concerning people, events, objects or other entities. Data can be financial and quantitative, or it can be qualitative and subjective. It can be internal or external, historical or predictive. There are many sources of data within an organization.
>
> Information is data that have been organized or prepared in a form that is suitable for decision making. For example, a list of students and examination marks in random order is data, but displayed in order of marks, highest first, it is information which enables a decision on 'who came top'.

This kind of distinction is made in many computing courses, but is generally dealt with in isolation from other parts of the course. In a database course this fundamental concept would be developed at the beginning, with the use of modelling techniques.

### 3.2. Data Models.

We are likely to collect information about anything of interest to us, a person, place, event, object, or organization. It is possible to model this and use the model in developing systems. This approach is rarely used in schools, but would be important in a database course.

In the realm of reality, we can distinguish entities (a things existence) and entity sets. We can establish a unique name for an entity and also establish attributes (values) for an entity. Only a part of reality may be viewed in this way, and there is always a trade off between flexibility and efficiency in a database system.

Having established what an entity is, a database is concerned with 'filing' representations of entity records, but more especially with the relationships between the entities. An understanding of entities, and the ability to represent them after analysis, is a key part of a structured analysis of a system for computerization.

Many school and college based courses are firmly established in the world of the machine, rather than in the world of information :

| WORLD OF REALITY | WORLD OF INFORMATION | WORLD OF MACHINE |
|---|---|---|
| Entity and properties | Entity record | Record |
| Entity type | Identity attribute | Key |
| Entity name | Entity identifier | Key value |
| Property class | Attribute | Data item |
| Property | Attribute value | Data item value |
| Entity set and properties | Entity record set | A collection of all the occurrences of the same record type - a file |

The next three sections look at how the teacher must move into the world of information.

### 3.3. Entity Relationships

These can exist in the context of any organization, examples would be:

invoice numbers
product codes and descriptions
sales and profits
employee number and department

This can be in the form of 1:1, 1:n or m:n representations. One to n relationships are hierarchial, where one record at a level is related to n records at a lower level. It is the easiest relationship to deal with and implement. M:N relationships are the most general type of relationship.

#### 3.3.1. Example Relationship

Entity Relationship Representations

To give an example from a real situation, an administrative task in a College is to coordinate all course marks for a set of courses. The marks are written onto Broadsheets. A set of courses will each contain students (no student may do more than one course. Each course is divided into a differing number of modules, and the modules will be a part of several different courses. A single module from any course is called a session, and has its associated lecturer. Each lecturer has a set of marks for the session. All of the session marks have to be collected together to create the course broadsheet. The Entities can be established as :

courses
modules
sessions
lecturers
session mark sets
students

The Entity relationships can be represented like this :

course : students 1:m

courses : modules m:m
this is split into two 1:m relations;
courses : sessions 1:m
sessions : modules m:1

marks : sessions m:1

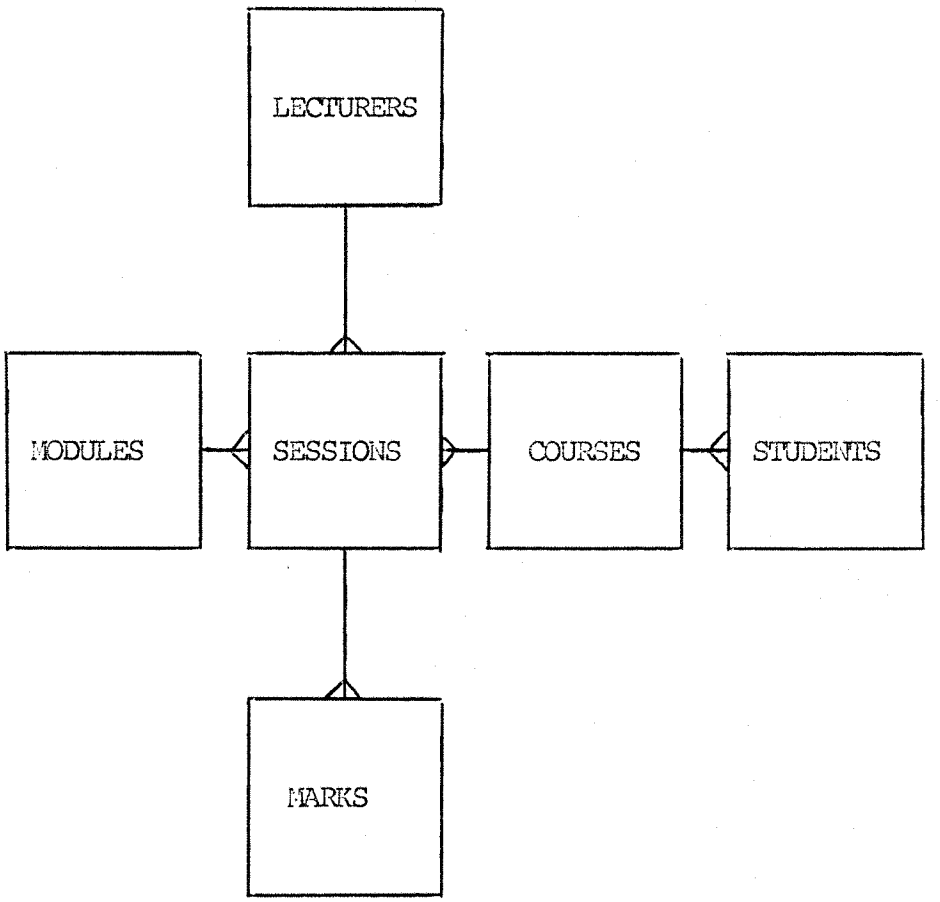



**Fig. 3-1: Entity Relationship Example**

The computing teacher will need to become familiar with this kind of analysis in order to teach the database course. Students do not

neccessarily have to understand the concept, provided the relationships studied are restricted to 1:m. In order to deal with complex systems this or similar modelling techniques will be needed.

### 3.4. Data Flow Diagrams

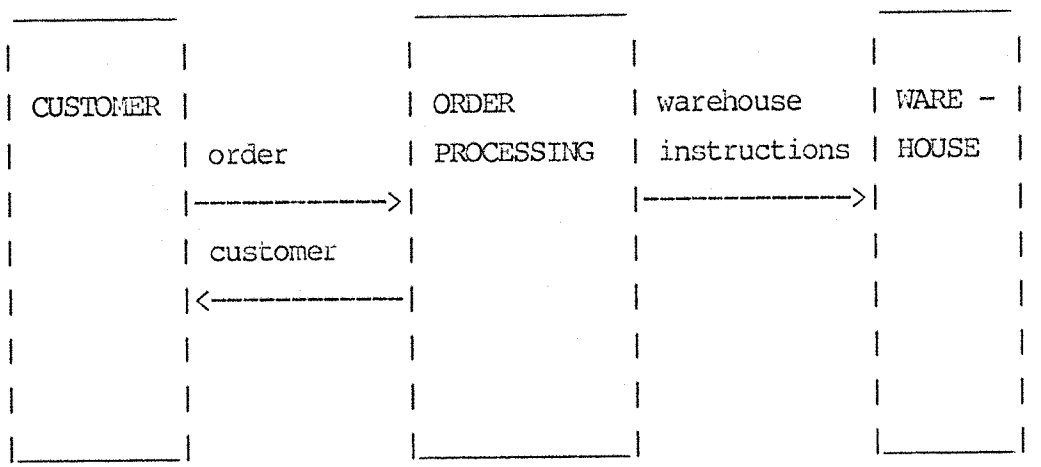
These are part of many structured systems methodologies. Many influential writers have propounded the technique, and there is much literature on the subject.

They can be used in different stages of the analysis, to represent the existing system (or aspects of it), to show documents files and forms, to indicate the logical view of the desired system, and to present different alternatives to the end user. It is therefore important that potential end users have seen such diagrams.

The best known textbook system is that of order processing, and this is a Data Flow Diagram for such a system (after a diagram from 3.3.,Hancox M 1985)

### 3.4.1. Data Flow In Order Processing

There are only four symbols to deal with, the elongated rectangle for anything external to the system, the smaller rectangle for any process involved in the system, an arrow showing the direction of flow of data, and the three sided symbol for a data store. Such diagrams are usually developed in a top down fashion, and the order processing system might fit into a wider system like this:




Each of the various processes can be decomposed until the 'bottom level' is considered to have been reached. Details of processing are not included, only an indication of data flows. Although the system is represented, the processes still have to be communicated, perhaps through psuedo code. The top down style is reinforced by level numbering of the diagrams.
For more detailed information on data flow diagrams the book by Gane and Sarson established the technique (3.4. Gane C & Sarson T 1978)

The compilation of such a diagram is an iterative process, and there are pitfalls, including:

The same datastore shown on separate diagrams can be confusing. Control and decisions are not included in the diagram and must be dealt with separately. Data flows can be easily missed. Superfluous data flows may be included.

In general, Data Flow Diagrams are a valuable tool in system communication, analysis and design. The technique is now included in the formal structured methods taught, for example SSADM. The technique taught in schools tends to be the older one using systems flowcharts. That schools will change to the structured techniques is probably a matter of time, and colleges are now in the process of doing this. The use of a database as the key element of a computing course means that these newer techniques will form an ideal vehicle for teaching students about systems. The teacher of a course using a database will need to adopt these newer techniques.

The diagrams are used to communicate ideas to the users of computing systems, and are designed to be easy to use and understand. It is reasonable to suggest that their simplicity will lend itself to teaching students as well.

### 3.5. Data Dictionaries

Data about data (sometimes called megadata) is a common software systems management tool which is frequently overlooked in computing courses. A data dictionary is an essential tool in the management of any system, however complex.

A data dictionary should be set up for any project in which a student is involved. The dictionary can be set up in a dictionary package, or can be set up for example, in dBASEII or any database package, or can be done by simply keying data into a unix file (through a wordprocessor) and using the unix grep or sort commands to access and arrange the data. There are commonly four basic fields of information stored :

item name
item type
cross reference (this is a repeating field)
item description

The item name should be unique, and so this can be used as the key to the file. The types of item would be (3.5 McFadden F R & Hoffer J A 1985)

| Structured Analysis Description | Equivalent to: |
|---|---|
| data element | field |
| data structure | record |
| data flow | file |
| data store | file / database |
| process | system / program / module |

The management of a data dictionary itself creates a problem in database design and implementation, and so is an ideal subject for student work. Any system or program produced by a student should include a data dictionary. The teacher using a database package can easily insist that students produce a formal automated data dictionary at least for the purpose of documentation.

### 3.6. Database Fundamentals

The thesis is suggesting that the use of a database management system will require the use of these analysis techniques, to some extent by the student. The teacher will need to understand and have experience in using them. The database itself will be studied in more detail than would currently be the case. Mosty computing courses consider databases to some extent, for example GCSE courses teach what a database is. Most students will be familiar with the definition of a database, they will also need to be familiar with the definition of a DBMS, like the ones quoted here:

> Date (3.6.,Date C J 1981 p.35) defines a Database Management System as the software that handles all access to the database, and sits between the physical database and the user and may rely on the operating system to provide access functions. On the other

hand he defines a database system as (Page 3) a computer based record keeping system: that is a system whose overall purpose is to record and maintain information.

An alternative definition is taken from (3.7. Deen S 1980) as follows:

> A generalized integrated collection of data which is structured on natural data relationships so that it provides all necessary access paths to each unit of data in order to fulfil the differing needs of all users.

Students would understand the concepts involved in these definitions through using the DBMS to solve problems, and the last definition from Deen illustrates the view that the students will have of the course, involving understanding what the users require and what the data relationships are.

### 3.7. The History of Computing

As a further demonstration of the way in which the emphasis of the course may change, it is worth considering the way that the history of computing is approached. This is a fascinating area of study, which students tend to appreciate since it relates to individuals (like Babbage) and to recent historical events (like the enigma machine useage in the second world war).

Most courses iclude computing history and involve aspects of the way in which database management systems have developed. School courses use the example provided by the U.S. Census Bureau in 1890 as an important part of the history sequence, since this represents the point in time when an organisation perceived that so much data was being collected it would only be possible to process it using a machine. Using punch cards (for the first time to store such data) the bureau created the first database which could be handled by mechanical means. It was a reflection of the filing cabinets and record cards that have always been widely used. Punch card files were

later used by computer equipment. Hollerith, the inventor of the system began the company which installed the equipment in the census bureau, and that company later became IBM. In database managment text books this is also an important part of the history of database systems. In other words the historical aspect of computing interlinks database development.

The historical viewpoint could focus much more on the database aspect, as well as on the hardware (generations) aspects which tends to dominate current school courses. For example, comparisons can be made with the same census bureau for the 1950 data, where magnetic tape useage was in many ways similar to the punch card system. File processing was still sequential and the terminology of files, records and fields was still applied.

Other inclusions could be :

As systems became more complex several programs shared files and integrated file systems became common. This did not imply any management of the shared files, but rather the access to the files through several different application programs.

As COBOL became widely used as the file processing language, it was soon apparent that a COPY statement (inserted into a COBOL program the compiler will Copy in file definitions - or other program elements) increased efficiency. This meant that a global data description was built up for any file system. Should any change be made to the structure of the records in a file, the source file definition needs to be changed, and all of the programs COPYing the definition need to be re-compiled. Otherwise any change in the file structure would need a source code change in every program. The result of using the copy statement is that someone needs to manage the central source definitions.

The IDS (Integrated Data Store) was introduced in 1965 and was

the first recognizable forerunnner of modern DBMS. It enabled large integrated file sharing by several users.

Creating a database that was program and language independent. In particular a change in the data should not require a change in an application program.

The CODASYL committee set up a Database Task Group, CODASYL DBTG which reported in 1971. In 1972 ANSI (The American Standards Institute) set up a working party to standardize databases. Parallel development was carried out by several manufacturers, especially IBM who produced two new models for future database designs. The IBM models were a Relational Database Model, and a Data Independence Accessing Model.

The Logical data requirements reflected the user's view of data entities and their relationships, and these could be described using a Data Description Language (DDL). This produces a map (called a schema) which effectively describes the entire database.

In 1970 Edgar Codd defined the terms now used for relational structures. These derived from mathematical ideas rather than practical commercial file handling concepts. The ideas revolved around mathematical concepts of relations. In a relational system: (3.8.,Codd E F 1970 pp377).

A file approximates to a relation
A record approximates to a tuple
A field approximates to a domain
Field types are attributes

From the late 1960's a number of people considered relations as the building blocks of databases (3.9. Childs D L 1968 and 3.10. Levein R E & Maron M E 1975). For a description of these see (3.11.,Date C J 1975). In 1970, E.F.Codd (3.12.,Codd E F 1970) of IBM proposed a model for a generalized relational database system, chiefly to provide

data independence and consistency (which are difficult to achieve in the formatted systems). The model was subsequently expanded and improved by Codd.

The history of computing could be presented with a larger database management content which students could relate to their own experiences of using and creating database systems. The view of the course could change to give more emphasis to database aspects

### 3.8. What Is A Relational Database?

This would need to feature as a part of the student course content. The teacher of the course would need a very good understanding of the concepts and special tools needed. The concept of a relational system is particularly suited to teaching novice users - this is why almost all microcomputer DBMS are relational. If the Ashton Tate view of relational systems is considered :

> First of all, a database is a systematized organization of computer files for central access, retrieval, and editing (3.13. Ashton Tate 1984). It is not just a collection of tables, but also includes the functions needed to manipulate the tables. In a Relational database, all the files look like tables. Some structures are rather complex, but the simplest files all look something like this simplified example:

Employee Roster

| | EMPNAME | EMPNO | EMPDEPT | EMPSAL | EMPSTART |
|---|---|---|---|---|---|
| REC1 | Allen, F. G. | 211 | ACCT | 18,000 | 01/23/79 |
| REC2 | Brown, H. B. | 354 | ADMN | 14,500 | 03/22/82 |
| REC3 | Calendor, P. | 134 | PERS | 21,000 | 12/17/77 |

It is clear that this concept is not only simple, it is ideal for use on school and college courses. The students are presented with a system that works in a way that is familiar to them. Ashton Tate emphasise that "A relational database can most easily be understood by describing the data and access paths as a table. This is so simple that at first it seems inadequate to meet the demands of any user." This approach seems ideal for learning about computing systems.

Robert Bowerman (3.14. Bowerman R 1983) considers that Relational Database Management Systems for microcomputers are particularly promising because they offer the user the ability to focus on the information required in order to do their jobs. From the user point of view the initial creation of the database is simple (compared to a formatted system). The life cycle of such a system is much faster than the traditional one, involving fewer steps. This is because a distinction is created between the function and the system interior. The systems top layer follows a well-behaved relational model, making it easier to understand and use. The bottom layer contains all the non-linear physical structures and is conveniently hidden from view. the top layer provides external appearances of logical data structures which can be called virtual views. It is this separation of physical and logical considerations that makes the RDBMS cycle possible. Because an RDBMS hides all implementation details it becomes possible to take the relational model of the current manual system, and simply tell the RDBMS what functions are required. It is the functions and not the design that is implemented using a RDBMS.

This is the most important aspect of the use of a DBMS in teaching computing. The students are freed from the lower level considerations (for example writing a sort program in BASIC) and can concentrate on using the system to process information. The emphasis from the student point of view changes to the higher levels discussed above.

A relation is a mathematical term for a two-dimensional table, characterized by rows and column. Technically called a relation rather than a table because the entries are homogeneous in columns but

not in rows. In a formatted database the structures are designed to meet the access requirements. These have to be correctly determined and may change. Changing the data structures is complex. Codd's relational model provides universal access paths and any data item can be retrieved with equal ease. This simplicity of these concept seems to make them ideal as the tool to use in teaching students about computers and computing systems.

The student does not need to use the relational terms, certainly not in dBASE, where the data processing terms are used in the package:

| Relational Terms | Data Processing Terms |
|---|---|
| relation | file |
| tuple | record |
| attribute | data item |
| degree | number of fields in the record |
| cardinality | number of records in the file |

Students would need to understand the differences which arise between relational systems and data processing systems. Although such differences may not appear because of the way in which the data processing application is designed students need to be familiar with the differences :

| Item | Relation | File |
|---|---|---|
| repeating group | not allowed | allowed |
| ordering of domains | not important | important |
| ordering of tuples | not important | important |
| duplicate tuple | not allowed | allowed but unlikely |

### 3.9. Normalization

Normalization is the process of reducing data to its simplest form,

thus producing the most effective structure for data storage, avoiding duplication. Many would argue that the technique of normalization should be followed when developing any system, but COBOL programmers, for example, would argue that the inherent disadvantages (from their point of view) in creating more files than are necessary will give rise to undue overheads on the implemented system. However, in a relational model normalization is needed. How would this affect the computing course ?

Normalization has been formally defined by Codd to consist of three distinct stages. There are two further stages (fourth and fifth normal form) which are also defined, but are less often used.

The objective of normalization is to reduce data to 'third normal form' which is achieved when all the repeating items are collected by key and no data item is identified by another data item which is not itself a primary key (3.15. Lancashire G 1985) Assuming that the data is already set out into 'unnormalized' tables, then there are three steps required to reduce the data to third normal form.

The teahcer will need to be aware of the techniques involved, and the use of entity relationship diagrams (described previously) as a cross check is useful. Since both techniques arrive at the same result I would suggest that students use entity relationship diagrams as a technique, rather than formal normalization, which is much harder to follow through. Some normalization concepts, such as removing repeating groups in first normal form, as simple enough to be arrived at through experience (directed by the teacher). In the next chapter classroom experience will suggest ways of handling this aspect of designing a database system.

### 3.10. SELECT FROM

Examples of a Relational Calculus system, as implemented, are the languages SEQUEL and QUEL. In general the syntax is of the form:

SELECT : lists the columns to be projected
FROM : identifies the tables used
WHERE : includes the conditions for tuple selection within a single table or between tables implicitly joined.

An example might be:

SELECT : EMPNAME
FROM : EMPLOYEES
WHERE : SALARY IS GREATER THAN 12000

Of course the syntax is much more complex and more powerful than the example given. The advent of the new IBM microcomputer system will utilize this syntax.

It is not the way in which dBASEII approaches the user, and this will be discussed in chapter five. It is the use by dBASE of its own programming language that creates the link between the database system and the 'normal' computing course content.

### 3.11. Summary

The thesis is viewing the use of DBMS from an educational viewpoint, and is using the success of the dBASE system with non specialist users as a guide to the correct educational approach. Using a DBMS will change the required knowledge and skills of the teacher, and to a lesser extent the student. There are 'wrong conceptual conclusions' to be drawn from using database packages without understanding the conceptual background. To quote four examples from dBASEIII:

The basic unit is always defined as a file rather than a table.

The user is required to know a lot about the physical database, for example that the index files are both seperate files and have a .NDX ending.

A view is defined as a 'main file' being used to access data from a chain of 'related' tables. This is only a 1:1 relationship and access can only be through an indexed field. This is much more like a COBOL ISAM system than a relational view.

A query is defined in terms of a 'filter' which is a conditional 'filtering out' of unwanted records, from any particular file.

The teacher does need to be aware of the fundamental concepts of relational systems in order to be able to interpret the package for the students, and also to be able to structure the way in which the package is used by the students to learn the required concepts and skills.

The emphasis of the course could change to give more emphasis to data and the world's of reality and information. The student will be able to focus at a much higher level than is currently the case, spending more time on systems aspects and less time on the lower level aspects of a computing course.

This discussion of the conceptual framework implications for an educational DBMS course raises points which are best examined in more detail in further chapters of the thesis, as other themes and evidence are discussed. One of these points, central to the thesis, is whether the change of emphasis would be advantageous for all computing courses. This is discussed in further chapters, and especially in the summary chapter (8) where the new GCSE computing course is discussed.

The next chapter looks at how these conceptual implications have been used on a group of students using dBASEII as a fundamental part of their computing course.

# 4. Chapter Four The Basingstoke Experience

The previous chapter looked at the change in emphasis and content that would be apparent in a DBMS course, together with the change in extent and level of the knowledge and skills required by the teacher. These changes reflect the move towards 'structured analysis' which should be common in schools and colleges eventually.

This chapter looks at the experiences of using dBASEII with a group of students following a BTEC National Computer Studies course at Basingstoke College, during the session 1985/86. The students were not taught in the way evisaged by the thesis, but their course involved elements of the DBMS approach. The response of the students (on a subjective basis) was very promising, and although few objective conclusions can be formed, the experience does suggest that a DBMS course will improve the student experience.

## 4.1. Background

This section discusses the background of the course, together with the reasons why the DBMS approach was introduced.

The course content has been discussed in chapter two. There are twin measures of success since the students will either find employment in a computer related field (trainee operators or programmers), or go onto higher education. In the session under discussion all of the students wished to find employement at the end of the course (and non changed their minds during the two year course). There were sixteen students on the course, which is a 'normal' size for such a group.

The students study six modules each year, which are designed to interrelate. Each module is taught by one, or perhaps two, lecturers. The course is controlled by a coordinator (this was the author) who tries to ensure that the work of all the module lecturers forms a coherent whole.

Courses change from year to year, and in this session the course was changing rapidly, in preperation for a new BTEC syllabus expected in 1987. Several problems had been recognised:

The need to organise methods of teachingstructured programming throughout the course. (Structured systems analysis had not been suggested).

There was a perceived requirement to move towards formal structured programming teaching, although most staff teaching programming had no background in structured methods. In general students were already taught to make their programs modular.

Moving away from BASIC in the first year of the student course.

This was not acted on since PASCAL was not yet available and this move was not supported by local businesses (where the students would hope to find employment). 'Business BASIC' was widely used locally.

The need for students to fully understand microcomputers and associated packages.

This was an increasing requirement from local employers.

Generally making the student aware of the problems of handling large amounts of data and organising work involving teams of people.

Commercial programming skills are an important part of the course, and BASIC has normally been taught from this point of view. However, it has proved very difficult to direct the way in which individual lecturers teach their respective courses, and BASIC is particularly susceptible to a bottom-up and unstructured approach. In order to overcome this the lecturers are asked to use a standard teaching system for BASIC which begins by getting students to design input and output, use DATA statements as

sequential files, and produce output to required constraints by reading through the DATA 'files'. However, in practice much time is spent at the lower level of programming, for example when developing skills in programming sort routines. Discussion has focussed on developing the use of LOGO and replacing BASIC with PASCAL. In practice this involves the lecturers in a lot of extra work, and so there is a built in inertia to BASIC systems.

There was an ever increasing percentage of students entering the course who had some expertise already in BASIC, either gained in school or at home. Many students were considered to have acquired 'bad habits'.

Second year students were placed with an employer for one day each week during the second year of the course. This produced a lot of employer feedback, and one of the key needs was to produce students who would present themselves as competent (potential) computer personnel at the start of the second year of the course. It was important for students to feel that they had handled a data processing assignement during the first year of the course. Experience in producing programs was limited to large BASIC programs which generally involved some kind of on-line file maintenance system, using the relative (direct) access facilities in BASIC. This was partly counter productive since students used ISAM systems in COBOL during the second year, and were not likely to meet such BASIC file systems on their placement. (The business BASIC systems use ISAM files).

Another constraint in the course arises from problems at the start of any college term. Since enrollment is only a week before the start of the term there are many timetabling changes during the first few weeks of college courses. It was therefore decided to use packages to teach the students during the first few weeks of their course. This had other benefits:

Students generally now had some exposure to BASIC and often did not appreciate that other skills, apart from programming, were important. Using packages would present the students with

problems other than programming ones.

The students could use packages to complete data processing exercises.

Students would be aware of what commercial programs look like and do. The packages discussed show them what to aim for when completing their own programs.

## 4.2. Approach

A good course introduction was required, that would provide practical experience for the students, and give them a good idea of what commercial computing involved.

In an attempt to both overide the BASIC inertia, and try a new approach, one group of students was exposed to some common packages (wordprocessor, spreadsheet, filing, accounts) at the start of the course, and then embarked on a major data processing exercise using dBASEII. This resulted in the students designing and implementing systems using the database package, and working as a team. Attention was focussed on the management problems, the input of data, and the required report outputs. The amount of dBASE programming required was small. The students were introduced to the idea of files - in an extensive manner, including the idea of indexes (which aren't available in BASIC). They also found the concepts of the WHILE - WEND loop, the IF - THEN - ELSE and the CASE structure introduced.

All this happened before the students were taught any traditional programming at all.

A simple case study was designed using a real information requirement. Within the various Departments in the College, there was no information available to the head of department on the origins of students. Detailed information would aid in marketing the college courses. Several requirements were defined:

A report showing which schools full-time students came from.

A report showing how many students came from particular firms.

A report showing which locations (areas) students came from, both generally and as specific groups.

These reports were to be general and also by class and by subject goups.

Up to date class lists for students (name and enrolment number)

The last requirement was for the purposes of the case study only. The source material was simply the enrolment sheets completed by course lecturers at the start of the year. These were hand written sheets compiled by lecturers and were often very hard to decipher.

This was a simple case study, with clearly defined input (the data sheets) and clearly defined output (the reports), with a user on site who had agreed to the scope and nature of the case study (the head of department - the case study was restircted to one department). In total some 900 records (student details) were involved. The success of the case study could be measured by the reaction from the user.

The data was to be collected using dBASEII. The students were initially taught how to create files, append and edit data, and generate reports. The CAL package dBASE Educational was also used.

After this initial exposure to the package, the students were split into teams of four, and a quarter of the data sheets given to each team. Each team was asked to discuss the planning of the case study work, and the following scheme was agreed:

Specification Review

Plan necessary files (including normalization).

Implement structure in dBASE II

(Plan codes for the industry / area classifications)

Design controls for the data file creation

Plan and implement the keying in of data

Check accuracy using controls

Use the report generator of dBASE II to produce the required reports

Key the report data into a graphics package to generate the bar graphs required by management.

The teams had to:

Coordinate with each other to make sure that their data could be merged into one database.

Plan for accurate and valid data entry.

Organise their work as a team.

The flexible nature of the package meant that the teams did not need to concentrate on the reports until data had been entered. In other words the accurate entry of data was the prime first requirement. The lecturers ensured that the correct file structures were chosen. The production of the reports was considered to be a relatively simple task once the data was entered.

It was an interesting experience for the students to work as a team, and agreeing on appropriate codes for area and employers involved some arguments. The codes could only be obtained after an initial manual

scan of the data sheets, and the students settled on using a three character code for the location, and also for the employer, with associated reference files.

After completing the case study the students began a shortenned BASIC programming course, but did use dBASEII at later stages of the course, notably to look at indexed files.

### 4.2.1. Normalization

No formal database teaching was given to the students, unless it was needed for the case study. There were no repeating groups in the data and the only normalization attempted was the creation of the 'reference' tables for employer, location, and school codes.

The students arrived at the need for these tables through validating the data entry, and it was interesting that they produced a normalized system. This was due in part to the system itself. Carefully directing the work of the students and choosing appropriate case studies should ensure that good database designs are created without formal normalization teaching.

## 4.3. Student Reaction

The students did not expect to have to work as a team. Coming from school they seemed to have little experience of this kind of work. The students found it a useful way of getting to know the other members of the group. All of the students started with the same degree of knowledge (that is very little) about how to organise the project. There was no advantage with those experienced in BASIC, as is often noticably the case at the beginning of other computing courses. Most of the students found learning the dBASEII package relatively easy, and lecturers gave a lot of help with this aspect - limited to entering data, editing it, and producing reports. Students were competent in these aspects of the package within two days.

The students reviewed the specifications through team meetings, and also decided on file structures. An overall discussion then fixed the structures for the group (all teams had to use the same file structure). The students were allowed to formulate their own data entry system, and chose to simply key in the data, print it out, and use a manual check against the original data sheets. Manual preparation involved coding locations and employers onto the data sheets. The students themselves keyed in the data.

Since the students had no control over the input documents this seemed a reasonable approach, and time did not allow for verification by entering the data twice. It would also have been neccessary to write a dBASE routine for verification and I wanted the students to control all aspects of their project. Writing a verification program would have required a lot of detailed dBASEII knowledge and would have overcomplicated an essentially simple case study. A later check by me found that there were very few errors in the data, and certainly not at a level to invalidate the reports.

One simple check could be used, involving the subgroup totals generated by the dBASEII report generator. The total number of students in each group was calculated manualy and entered onto the data sheets. Reports generating the same totals allowed later checking to correct errors. This simple technique ensured accuracy in the student entry in class lists reports.

None of the student groups completed the work in the given deadline and extensions were given, in all the project spanned some four weeks. This was much longer than either the students or the lecturers had anticipated. In the last week some programming using dBASEII was introduced in order to produce reports beyond the scope of the generator.

In general the students learned a lot about :

each other

the problems and advantages of organising groups

organising data collection

the fact that even simplistic data processing tasks are difficult to organise and manage

the package dBASEII

Sequential files

producing output to a deadline and to the specification of a user

using a computer system to process data in a realistic way

Many of the lessons of this approach are subtle. One important lesson learnt by the students at a very early stage comes from using dBASE II. They can soon begin to design and implement a data processing project using large amounts of data and working as a team. Even those among them who are seasoned BASIC enthusiasts soon realize that the problems of managing a team and managing large amounts of data are much more complex and interesting than those involved in writing programs as individuals. Team programming used later in the course benefited a lot from this early exposure to teamwork.

### 4.4. Results with reference to BASIC

The students then had a shortened period of training in BASIC, before entering the second year of the course. The BASIC work proved more superficial than usual because of time restraints, and it was noticeable that from a time point of view, the students spent a lot of time in BASIC sessions, with little learning reward, whereas they had learnt much in the dBASE sessions with a large knowledge / skill gain in a much shorter timespan.

A key finding is that the students began to question the validity of BASIC. They came to see the language as a tool for learning some narrowly defined learning objectives - for example when discussing sort algorithms (and BASIC programs) with one lecturer. They did not perceive BASIC as a tool for using the computer to do realistic jobs. This was not a viewpoint deliberately fostered by the lecturers. In many ways the students had a 'mature' view of the value of BASIC in their studies. BASIC had a clear role which could be distinguished by the student from the role of the dBASEII package and other parts of their course. They were able to view the course much more as a consistent whole, not focus too much attention and time on BASIC for its own sake.

The BASIC course followed by the students was designed from an american system, using BASIC to handle 'files' of data held as DATA statements (to begin with). The starting concepts are simply:

```
LOOP
     READ item1,item2,item3...
     PRINT FORMATTED item1,item2,item3....
ENDLOOP

DATA item1,item2,item3.....
DATA ........
```

All of the students in the group found the course simple, are there were noticably no students who got 'lost' at any stage in the BASIC course. All of the students emerged from the BASIC course feeling confident about their use of the language. Other lecturers used the language in other modules to demonstrate concepts and for assignment work.

The group were also subjected to multichoice tests on all aspects of the BASIC course. These were machine marked (using a unix system) and all students scored consistently over 80% in these tests.

## 4.5. Results with reference to COBOL

From the course point of view, most students will find employment as trainee programmers in a commercial environment, and so there is an emphasis on COBOL, which is the most common local requirement. RPG II/III is a growing need, and students are also taught unix and C to some extent. They will also cover assembler languages. This will equip them for a full range of employment prospects as trainees. However, we have found through past experience that employers look for a high level of knowledge and experience in potential employees, and will choose those students who can demonstrate this. Employers are not necessarily prepared to train their programmers from the level that might be expected from the course, and this means that the students aim to reach a level of competency higher than required for the BTEC board.

The observed effect was that the students began to think in terms of programming as a means to producing the required output from the defined input, and for a particular management need. They had also absorbed the concepts outlined without any particular formal teaching, and this understanding stayed with the students throughout the course. Normal practice, which involves introducing file structures at the end of a BASIC course which only allows the implementation of sequential files, has been found to produce a poor understanding of file organizations after much lecturer input. This seems to be a case of learning by doing being better than learning by listening.

During the second year of their course the students were found to have become competent COBOL programmers in a short length of time, and indeed invariably, and throughout the group, produced totally reliable working programs which were fully structured, and produced on time. Of course, it is not possible to credit this to the dBASE II experience since there was no control group. However, I do consider that the student dBASEII experience was very influential. The students produce well written, structured programs that work with

little (and often) no debugging. This has never been experienced in the past. Of course there may have been a flaw in previous BASIC/COBOL teaching techniques.

My own view is that the students were able to gain from their use of dBASEII. In that package the student was able to effectively take a high level view of the system and write programs in dBASEII which were like the psuedo code designs for their COBOL programs. This is because the language is at a higher level. Further they had an excellent grasp of files gained when using dBASEII.

It was not only on the programming modules of the course that the students did well, their performance was good across the range of modules that they took.

### 4.6. The Job Placement View

Two of the students did not perform well on their placements, but this was considered to be due to their own personalities and shortcomings that any particular college influence. Most noticable was the good performance of the 'average' student. Those students who are 'good' at programming will generally do well on their placements. The main effect noticed was that students who were considered 'average' or worse did noticably well on their placement.

Employers generally were satisfied with the students and the skills and knowledge which they showed. Several expressed an interest in the fact that dBASEII was used as well as BASIC during the first year of the course, although they were generally more interested in the use of COBOL during the second year.

### 4.7. The Coordinator's View

The students had learned valuable lessons in managing the problems involved in a systems project, and working as a team. All following courses in programming emphasized and expanded on these aspects, and

the students became quite used to working from this point of view, rather than in considering the program as an end in itself, or as a 'piece of art'.

In support of the success of dBASE II with this particular group of students these factors may be quoted :

The group obtained the highest number (and percentage) of BTEC distinctions ever obtained by a group at Basingstoke.

The group have all found employment, many as trainee programmers, in the local area.

The group of students who are currently in their final year at Basingstoke, and who did not have any dBASE input, have fared much worse on their COBOL course. They do not have the depth of knowledge of files or program structure that the dBASE students had, and they are perceived as being 'not as good' as the previous 'dBASEers'. It should be said that all kinds of other factors may account for this. Most noticably the group have not adopted to carefully designing their programs, as did the previous group.

In general the way in which the dBASE students designed their programs was the most noticably improved aspect of their work, together with their increased skills in working within a team. The performance of the 'average' student on programming courses seemed to improve noticably.

### 4.8. Summary

The BTEC board have recently been consulted about which language they prefer to see incorporated in their programming modules. They have stated that whilst they do not impose any particular language constraints, they are concerned that a structured approach is used (the Jackson Structured approach being one example). BTEC have given

tacit approval to the use of dBASE II as the programming language (instead of BASIC) on this basis.

This brief summary of current BTEC course teaching practice revolves around programming concepts, because the aim of the course is to produce trainee programmers and operators. Of the students leaving the course most have found employment as trainee programmers, some as operators. Currently the most 'successful' students from the leavers are earning five figure salaries. I feel that the improvements which seemed to occur within this group of students should occur on students of any computing course.

The actual form of the case study can be easily varied, and provides students with many skills, such as those concerned with team membership, team leadership, and even project leadership. The students benefit greatly by handling large amounts of data, and learn all about verifying and validating data entry, and managing and controlling a large amount of data. This kind of experience is lacking in many courses because the emphasis on program development (at the lower level), means that students only test their programs with small amounts of data, and leave their course experience with little or no idea of the problems of handling large volumes of data.

From the lecturer's point of view almost any raw data can be used, as suits the students, because the database system does not limit the design process, as would such a system geared to an application package. The students have to provide all of the skills and control systems, which can be easily incorporated into the database system. Experience with several groups at Basingstoke shows that the students benefit when this case study is introduced early in the course. Students begin to understand the problems associated with handling large amounts of data, and the careful professional attitude that needs to be adopted if the data processing is to succeed. Many useful concepts can also be introduced including :

Files

Data Validation
Data Control
Team Management
Central Data Dictionary

Chapter three left the question of whether the conceptual framework of a DBMS course would lead to an advantageous change of emphasis on any computing course. The Basingstoke experience indicates that for this particular BTEC course the change was an advantage. In further chapters of the thesis I will be able to discuss this question in the wider context of other computing courses. However, since the core part of the BTEC computing course is similar to the core parts of most computing courses (for example O and A level courses), then the question has been tentatively answered.

This chapter has looked at one particular aspect of a course using dBASEII. It is not the case that a DBMS course is based entirely on the particular package used. As this example has shown the case study is as important as the package. The next chapter takes up this theme and considers the tools that must accompany the DBMS package.

## 5. Chapter Five - The Tools For Database Teaching

The previous chapter described experience of using dBASEII in a computing course. Emphasising a DBMS rather than a programming language, changes the way that the teacher and student views the computing course. The DBMS itself should not be the focus of student attention since this will create problems similar to focusing on BASIC as a major aspect in a computing course. Rather there are a range of tools available to the computing teacher, of which the DBMS is one, and which integrate the learning experience of the student. The DBMS is not the only tool which should be used in the course proposed by the thesis. There are a range of tools which can be used together with the DBMS. This chapter looks at these tools and the way in which they can be used.

### 5.1. Microcomputer Experience

Microcomputers are the most commonly used tool that students use to acquire skills in computing, particularly at the school and college level. Database packages may have a more limited scope when applied to microcomputers, as when applied to minicomputers (or mainframes). A commercial user of a microcomputer may happily claim to be using a database package, when a minicomputer DBA would not recognize the package as more than a filing package (a 'flat file' package).

It may not be correct to make the assumption that the presence of a microcomputer will lead to any kind of useful experience for the students who use it. As a tool the microcomputer can be readily abused. Filing packages, on the other hand seem to be readily assimilated and usefully used by students and teachers alike.

Apart from the articles discussed in the previous chapter, there seems to have been little investigation of what schools (and colleges) are using their equipment for. Although many schools are gradually acquiring more equipment, until very recently only one or two microcomputers would be used within a school. So far as schools are

concerned the most common area of use is with specialist (CSE or GCE) computer studies courses, and most of the computer usage here is concerned with the projects required from students for examinations. The main problem is with the limited number of machines that are available.

Other uses (broadly classified as either 'awareness' or CAL) are definitely limited by the lack of machines. In secondary schools most teachers do the same thing with the entire class (around thirty students) from lesson to lesson. There is generally little scope for teamwork, or individual work. An exception can be found in the remedial department, which is found in most schools and deals with, for example, children with low reading ages. Individual study is often encouraged in these groups, and so the use of perhaps one machine, by a rotating set of students, using CAL (often 'drill' type programs) is common.

In other areas of the school CAL packages are not likely to be successful because only one student can use the package at a time, and also because of the lack of training and awareness of the teacher. In a school where the resource is limited, teachers other than the specialist computing course teacher will find it difficult to get access to the machines.

Primary schools, although varying in teaching approaches, have always been ahead of secondary schools in inovative teaching techniques. It is often the case that children are encouraged to work at their own pace, and in a mixed ability grouping. In this environment the use of one computer can more easily be tailored into the teaching day, and the students rotate around the machine, perhaps using different packages. Pauline Bleach (5.1. Bleach P 1986) has recently published a study of computer use in 537 primary schools (this was a 77% response to a postal questionnaire). She found that the microcomputers were underused, and the software used was generally of the drill type, for example spelling programs (70% used) and alphabet skills (58% used). Wordprocessors were used by only 23% of schools

and databases (filing systems) even less. She was disappointed that the better packages, which develop skills in logic or imagination were not being used at all. These include writer an introductory wordprocessor package, developing tray which develops cohesion of text and mallory where children create characters and scenes.

There is, then, some use of CAL packages, a strong interest in primary schools in the potential of such uses, and a limited use in secondary schools where the equipment tends to be held by the specialist teachers for their own courses.

## 5.2. Flat File Packages

There are many flat file packages available, those commonly used include Vu-File and Quest which run on the BBC B machine. Quest will also run on the Research Machines 480Z, as well as the RM Nimbus and so is available on the machines most commonly found in schools and colleges. These packages have been used in various course areas and the students who experience their use are learning much about computers. The Quest package has not been reported used by Computer Science teachers, rather by teachers of other disciplines who are choosing to use the computer as a tool for analysis. The package comes complete with population census data of a village, and this has proved popular with Geography teachers.

A good example of the package use is reported in Educational Computing (5.2. Mendelson M 1985) and concerns the projects run by Michael Mendelson in a primary school. The Quest package was used to collate data about a local small river, and fields included depth and speed of the river, rainfall and wind, and also details of wildlife spotted. A record constituted a visit to the river. Having collected the data and keyed it into the computer (a worthwhile educational experience in itself), the children could then run various statistical analyses to produce information in graphical format, for example of rainfall over any period shown as a bar chart.

Mr. Mendelson seems well satisfied with the enthusiastic response of the children, and the results obtained. This kind of computer use is teaching the children much more about the way in which to use computers than might be taught in a Computer Science class.

### 5.3. The Langauge As The Tool.

It is worth discussing at this point the BASIC language of the microcomputers used in schools and colleges. It is this language which has so often formed the main (and sometimes the only) tool which the computing teacher uses with computing students. It is only in recent years that packages have been introduced into computer studies classrooms generally.

The actual hardware, whilst important, is not really the important aspect when using the microcomputer as a tool. It is the software that really maters. Many teacher users have tended to see a use for the hardware and make the best of what is available (which normally means writing a BASIC program) to create the software to make the machine useable.

Most school teachers use worksheets to create practical work for the students from their lessons. This is because most teachers prefer to create their own (or at least add to) workschemes rather than rely on those in books. There are two reasons for this, the first is one of professional pride in creating a tailored scheme of work, the second is that books are a scarce resource, and often not in parallel with examination requirements. In effect this means that teachers tend to create their own style of course.

This high level of input, as compared with for example the American model - where teachers may follow a detailed scheme for any particular course, is taken for granted. When given a microcomputer to use many teachers will expect to have to design a course around it. Computer Studies teachers did not see any anomoly in being given the hardware

and a language interpreter as their tool.

This and other factors have led to a large diversity in the way in which machines have been applied in schools. One of the other factors is the (almost) total lack of support or direction from central county groups.

This is against a background where the Government effectively reduced the state schools to two hardware choices - the BBC Model B with tape drives, or the Research Machines 380Z with disc drives. Yet the focus for this purchase was on hardware and not on software grounds - no central supply of software was really established.

### 5.3.1. MEP

The Microelectronics in Education Program which was set up to coordinate the micros in schools scheme, has recently come to the end of its scheduled five year life. This government backed program was responsible for funding a significant amount of software development, and MEP helped to develop some 80% of the educational software now available. This software was not simply specific to computer studies teaching, but was aimed at the broader curriculum.

The fact that only 2% of current annual capital expenditure is on software demonstrates the need for this government backing for software development and production. It is unlikely that commercial firms will spend money on developing software for such a low spending market.

The MEP have lulled schools into thinking that software - which they subsidised - will always be cheap. The article by Jenny Miller on the demise of the MEP (5.3. Mill J 1986) points to a new direction in software development. With no central software development, the major software publishers are looking not just at the schools education market, but at the home user as well. The forecast is that home users will start to require educational packages, and that this will provide

a new and viable market. This is being promoted by a group of educational publishers named BESA (British Educational Software Association).

The software component of the microcomputer is thus generally under resourced and of limited use.

### 5.4. DBMS as tools

Many teachers of computing courses now make some use of database packages, although in schools this is usually confined to 'flat-file' systems.

Database Management systems are rarely found in schools, partly because of their relatively high cost. Of those available dBASE II is the cheapest (at around one hundred pounds), but this will only run on CP/M or MSDOS machines, which in schools would generally be a Research Machines 480 Z. Schools recently purchasing RM Nimbus networks will obtain a bundled package called SUPERFILE. Those with BBC machines have no real DBMS option.

The thesis is suggesting that the language of the machine being used for the computing course become much less important, and a DBMS replace the central role that the language has held. Many of the learning objectives of computing courses are not really satisfied by using a language, and often the course seperates into theory and programming. This should not be the case, practical skills should be taught along with theory.

There are a lot of DBMS of which dBASEII is a representative. The following table taken from (5.4 McFadden F & Hoffer J A, 1985) and expanded, summerises some available systems:

Relational Algebra Systems

| Package | Vendor | Equipment | Comments |
|---|---|---|---|
| RIM | Boeing Commerical Airplane Company | PRIME 750 | Host language interfaces to FORTRAN, PASCAL, and COBOL. Typical algebra operators. |
| dBASEII | Ashton-Tate | Variety of micros under CP/M and MS/DOS | special procedural language as well as typical algebra operators |

Relational Calculus Systems

| Package | Vendor | Equipment | Comments |
|---|---|---|---|
| SQL/DS | IBM | S/730,3033; DOS/VSE operating system | Host language interfaces to COBOL,PL/1, and assembler;logical views supported;SEQUEL. |
| INGRES | Relational Technology | DEC VAX-11 VMS and UNIX | Host language interfaces to C, PASCAL, FORTRAN, BASIC and COBOL;logical views supported;QUEL |
| ORACLE | Relational Software | DEC PDP-11 VAX (VMS, UNIX, RSX RSTS) and IBM S/370 and 3033 (VM/CMS) | Host language interfaces COBOL, PL/1, FORTRAN, C, BASIC, and assembler; logical views supported; SEQUEL |

| Package | Vendor | Equipment | Comments |
|---|---|---|---|
| Knowledge Man | Micro Data Base Systems | Various micros under MS/DOS | QUEL-like query language and special procedural language; query commands may be embedded in simple spreadsheet cells; views not supported. |

Graphical / Tabular Systems

| | | | |
|---|---|---|---|
| Query-by-Example | IBM | S/370 and 3033 | Unique graphical fill-in-the-blanks query language |

Relational-like

| Package | Vendor | Equipment | Comments |
|---|---|---|---|
| NOMAD | National CCS | IBM S/370 and others | Hierarchial and relational models; limited logical views; special query language. |
| ADABAS | Software AG | IBM S/370 and others | Inverted file organization with some network constructs; several query and reportwriter languages. |
| REVELATION | COSMOS | Various micros | Menu driven PICK family system; data dictionary; network capability. Variable length fields and repeating fields available. Interfaces to BASIC |

| Package | Vendor | Equipment | Comments |
|---|---|---|---|
| SUPERFILE | SOUTHDATA | Various micros and UNIX | A relational system can be built in one file and fields can repeat and are variable length. Interfaces to BASIC and PASCAL. Little support from query language. |

The exact classification of data bases is not necessarily straightforward. The IBM IMS/DL1 system for example is called hierarchial, although does not qualify under the strict definition. It is called hierarchial because this best describes the system. A better way of considering databases might be to think in terms of relational simply meaning non-procedural. It is only necessary for the user to specify the required result and the software works out the required steps. Procedural, on the other hand, means that the user needs to define the steps in order to get the results required. The CODASYL system can be thought of as an extension of the COBOL language in order to allow a programmer to utilize a database.

It seems unlikely that many of these systems could be used in schools or colleges since they are expensive. However Oracle and Ingress are now available in PC forms. A brief description of two systems which could be used in schools is given below, as a guide to the diversity of available systems.

### 5.4.1. SUPERFILE

This package is from SOUTHDATA and is an interesting and powerful database package. It does not follow any recognised pattern (ie it cannot be easily classified) but can be used to set up a relational system - within one file. Available as a multiuser system on the NIMBUS network (The NIMBUS network has been a popular buy for schools and colleges this year) it will certainly be used because it is

available.

Unfortunately it has many raw edges, and would be impossible for a novice user to attempt. The 'user interface' is complicated and a high level of expertise and pre-use teaching is required. All of the fields are variable length and by default the system will index on every field. This means that an update can take around twenty seconds.

The manual which comes with the system is very poor, and any teacher would have to spend a lot of time creating teaching material. The way in which SOUTHDATA describe the package has little relation to text book concepts. For example fields (domains) are called TAGs, and a TAG may or may not be present in any record. This is not the same as a null field, the field may appear to be 'missing' completely.

This means that to create a relational system, the user defines a record structure as consisting of all of the TAGs for the entire system. A screen forms design utility (FORMS2) can be used to define screen input/query forms which could match, for example, the TAGs in one table. From the user's point of view, although they are using only one file, there appear to be several different tables. The problem with this utility is that although the fields are variable length, the screen entry restricts them to fixed length and so negates the underlying package. There is a report generation package as well, but like the FORMS2 package it is very awkward to use.

There is a very limited DML with the package, but all manipulation would need to be done through either BASIC or PASCAL. This could be promising, but the RM PASCAL is rather weak and the interface with SUPERFILE is almost so awkward as to be unusable from a student's point of view. Indeed, at Basingstoke several lecturers were unable to make the interface function at all.

The package is 'free' with the machine, and will be used by those who have it. It is a very 'non-standard' system and if used on a

computing course for teaching would demonstrate few of the salient 'text book' points, and would confuse the students who used it.

The package is interesting because it raises the idea of a microcomputer DBMS which could be designed to be easy to use and would support data entry and queries, but would then (unlike dBASE) interface to a language like PASCAL for data manipulation. Such a system could have great potential if suitably presented to the educational user.

### 5.4.2. REVELATION

Marketed by a firm called COSMOS, who claim that it is currently the best selling American package, this is also a LAN software package which will run across any MSNET system (like the RM NIMBUS Network). Part of the PICK family, REVELATION is designed to be easy to use and comes with a tutorial package. A menu driven system development package can be used, or REVELATION can be used directly in a PICK like way. A fully structured version of BASIC can be used to access any database and includes extended commands, for example the PICK LIST command.

Most potential users have not yet come across the interesting concept of variable length records, together with direct (Random) access through key field indexes - not a concept readily found in textbooks on DBMS. The system can be used as a relational model, or with repeating fields. REVELATION offers variable length records with repeating fields, allowing the user to build standard screens which easily allow variable length data to be input through windows. Complete with a menu driven system that any 'competent' end user could cope with after a little training. REVELATION has the strength that it can be used either by a competent systems developer using PICK commands (the REVELATION command level), or it can be used in a menu driven form by an end user.

Because the system has migrated down from mini systems, it would seem

to be an ideal micro system for database development. Complete with networking (over many different network systems) including record locking, REVELATION is a package that will allow networked microsystems to challenge the minicomputer / 4GL market. Powerful enquiry facilities are available with the PICK list command, allowing English like sentences such as :

```
SORT THE INVOICES WITH AN INVOICE.BALANCE GT 0 BY
CUSTOMER.NAME AGE BREAK ON CUSTOMER.NAME
```

This will produce a sorted list of customer invoices with subtotals for each customer (the BREAK ON phrase).

The user system design menu offers these functions:

- Define and Create a File
- Build a Dictionary for a File
- Set up a New Program
- Select Fields to be Displayed
- Generate a Standard Screen
- Customize an Existing Entry Screen
- Enter Data Using a Program
- Produce Application Documentation
- Generate RBASIC code
- Build a Menu
- Return to REVELATION Command Level

Revelation offers a good alternative to dBASEIII, and is in many ways better. The problem from an educational point of view is that the system is not 'text book' standard, and it is also little known. The teaching support is available, there is a tutorial package and a reasonable manual, but this is much inferior to the dBASE support. I would say that the strong point is the interface to a fully structured (PICK) BASIC which is directly applicable to educational courses.

### 5.4.3. Summary

There are good Database Management systems which offer alternative approaches to dBASE, two of which are briefly outlined above. These details were taken in part from (5.5 Hitchman S 1986). All have some strong points, and many weak points, dBASE included. When the thesis work was begun the dBASE II package was the strongest contender. The next section discusses why.

## 5.5. Why dBASE II ?

Of relational databases it has been often said that "Everybody wants one, but few people know what it is". (5.6. Sweet F 1984). dBASE II is one of the best known of the microcomputer relational database systems. It is widely used and accepted in the business and commercial environment and whilst it has associated disadvantages it would seem to be a good initial choice for a database system. This section points to the advantages and disadvantages which have been found through experience of its use.

dBASE II was the best seller and most popular software development tool available for microcomputers. So states Mr. Piper (5.7. Piper B. 1984). So far as can be ascertained locally (in Basingstoke) and in magazine articles this would seem to be generally accepted, and dBASE II has been a best selling package on numbers sold. In 1984 selling around 2,000 units a month, dBASE II has an interesting history. The author of dBASE II, Wayne Ratcliffe, started work in 1978 to help him complete his football pools. He produced a package called VULCAN. This was marketed and improved by George Tate as dBASE II. Released on the U.K. market in 1982, the British office now employs around 30 staff. Essentially it is designed to:

1. Create,maintain and store a large data file.

With an in-built programming language used for access and manipulating the data by stringing together english like commands into command files (programs).

2. Be more understandable than BASIC or PASCAL.

Apart from the common database facilities for searching sorting adding and deleting records, dBASE II has some useful arithmetic functions which make it especially useful for financial applications. The package does not have a floating decimal point which limits its use in scientific work. A screen layout generator is also available, together with facilities for indexing searching sorting and joining databases.However, the file size is limited to around 65,000 records, the user can only have two files open concurrently, and the sort facility is slow. There is a satisfied and large user base has proved a more successful marketing ploy than promoting leading edge technology.

There are several enhancement packages available for dBASE II including:

QUICKCODE - a program generator.
dGRAPH - produces graphs from database information
dUTIL - a software tool
SCRUNCH - a software tool
dBRuntime - a runtime package for application systems.
COMPANIO - a programmers 'companion' package which aids program design and documentation.
GEN - A program genberator.

## 5.6. Advantages of dBASE II

It is not the intention to assert that dBASE II is better than other micro database packages, either in the commercial or the educational environment. dBASE II is used because :

A. It is commercially widely used and perceived by such users to be a 'good' package. (This may reflect clever marketing by Ashton Tate).

B. dBASE II is available in schools at very low cost on hardware presently installed.

C. dBASE II has extensive on-line help facilities and an associated teaching package (called dBASE Educational) which is

also an on-line package.

D. There is a large library of books available concerning dBASE II and its commercial applications.

E. It has been used successfully in an educational environment.

F. Uses a procedural language similar in aspects to both COBOL and BASIC.

It is for these reasons that dBASE II has been used throughout as the database package example. During the course of the thesis writing several new (to me) packages have been experienced, and include : dBASEIII, Revelation, Paradox, Oracle. These are still rather expensive in educational terms, but are all superior to dBASE II. The concept of using any relational system can be established, if the possibilities for dBASEII are established.

Ashton Tate claim that "to anyone who has programmed in BASIC or COBOL, it will be clear from the introductory lesson that using dBASE II it is possible to achieve a great deal, in the way of file handling, for very little time overhead, and can be achieved by essentially 'non-programmers'". In the context of a microcomputer, where the cost of developing software must be kept to a minimal level, this claim holds good. Compared with both BASIC and COBOL, dBASE II is easier to use, and produces quick and effective results for any file based problem, provided that the problem is well defined and of the 'flat-file' variety.

## 5.7. Disadvantages.

There are rather more of these than the list of advantages, and they are based on experience in use.

1. Perhaps the main problem is the limit of two files open at a time. This does limit the extent to which a reasonable attempt to develop a relational system can be implemented. In use this has proved to be a drawback several times. However, it should be said that many users of such packages are interested in the 'flat-file'

problem, and normalisation of such a file may not be important to them.

When set against the results that students can produce using BASIC (or indeed COBOL) on a typical educational course, then it is still true that more can be achieved with greater emphasis on design using dBASE II.

2. The package has not proved easy for a novice user to feel comfortable with, until perhaps two or three hours exposure have been given. This is compared to specialised 'flat-file' systems (like SMART for example) which do not claim the intitial complexity of dBASE II and consequently gain on the ease of use aspect.

With students following a specialist computer based course this does not prove a great drawback, the complexity merits the initial user problems. For students on a more general appreciation course it seems preferable to use other less complex systems.

The comparison with BASIC here would suggest that the environment and ease of use (to solve problems) is much better with dBASE II.

3. Other dBASE II system limits (for example of 64,000 records) are of no consequence to the educational user.

4. Whilst a great deal can be achieved without resorting to the command file - application program - language of dBASE II this becomes inevitable on any extended case study. dBASE II has its own style of language here, and although this has many good points, it is not a standard in any sense. The approach is of the Relational Algebra type, and students have found it easy to use provided that the command files do not become too complex. This aspect is discussed further in the next chapter. There is a danger that students will develop 'bad habits'.

5. The CAL package (dBASE Educational) is generally very good, and

well worth using. However, there are certain parts of the tutorial sessions that are flawed, and bring most students to a sudden stop. Fortunately many of these problems can be cured by editing the command files (dBASE Educational is written in dBASE II), and this has been done by the author in several places. Some of the flaws were so obviously unacceptable to a novice user, that it does indicate a lack of user testing on the part of Ashton Tate.

On balance, the availability of dBASE II to those schools and colleges with either CP/M or MSDOS machines means that this package, rather than the newer better ones, will continue to be used for some time. All of this can be best demonstrated not by looking at dBASE II in detail, but by looking at how it can be used. To do this the next section looks at the use of case studies (another tool) which can include dBASE II.

### 5.8. Using A Case Study

The case study is a valuable tool in any computer based course. Computing is about applying technology to solve a problem. Students understand this best when they do it for themselves. This chapter is intended to exemplify the approaches to a computing course that can be made using both a database package (dBASE II) and a case study. The section discusses four case studies comprising:

1. Data Processing Exercise

2. British Telecom (low level)

3. Estate Agency (medium)

4. Administration Broadsheet system (high level)
   Detailed in appendix D

All of the case studies were developed and used by the author.

### 5.8.1. Data Processing Case Study

This case study was described in the previous chapter. It is included here for completeness. Any large data processing exercise, when presented as a case study is of value to computing students.

### 5.8.2. The British Telecom System

This is a simple case study taken from an article in the Times newspaper published in 1983. Rather an old article, but Telecom are still in the process of implementing the new billing system, and so the case study can still be used usefully. The case study concerns the current billing system used by British Telecom, which revolves around photographing users meters, and keying the information into the computer system. The Telecom system X which will eventually replace the older system nationwide, is also described. The case study has been used widely since 1983 by several lecturers, and successfully.

This is a popular case study amongst all students tried, since they can all relate to phones, phone bills, and the systems described. The students can build on their understanding of the system.

The case study would begin with the students reading the article. Consideration could also be given to the example phone bill, which is a real example of an incorrect bill. The students are then asked to discuss the nature of the Telecom system, both past and proposed, and would normally arrive at an image of the systems exemplified by diagrams. The required knowledge to understand the system would be :

how data is keyed in at a VDU
a batch as opposed to a 'real-time' system
transactions and master file concepts
storage media / concepts
validation / verification
processing data (including calculations) to produce output

However, it is reasonable to suggest (from experience) that most students can relate to the outlined system, and will assimilate the required knowledge in the context of the system studied.

#### 5.8.2.1. System Implementation.

At this stage of the case study, two things would normally happen:

The student would be left to guess at the computer system attributes.

The student will run a ready written in-house simulation program (for example on the BBC computer).

This is, of course, what would happen on a normal computing course, and the chances are that the lecturer teaching the course will have neither the time nor the determination to complete the simulation package. This case study has been used successfully with a BASIC package which produced fixed and simplistic bills.

In the system developed in BASIC, it was observed that the students gained a great deal by using the simulation program, in order to key in some transactions and see the 'bill' displayed on the screen. In particular there was some fascination with the idea that the computer could retrieve the name and address information simply on the input of the phone number (key field).

The problem is that the students have a very limited experience of the computer system, and cannot see 'behind' the system. They tend to treat is rather as they would a 'magic act'. A further problem is the inflexibility of the system, which is fixed into a BASIC program and not easily changed. The design aspect of the system is definitely lost, since the student is presented with an application package, and has to do no work in order to produce the solution. With dBASE II the student can be asked to design and implement a crude billing system and thereby learn much more.

In a DBMS course the lecturer can now place the students in the position of themselves having to implement the system for master file creation, and maintenance (simple enough in dBASEII) and can go onto develop the transaction update system. The advantage here is that the students can create the files and data and so become much more involved with the system and its detailed design. The system developed can thus be flexible, and is not transparent to the student, as would be the case if they simply used an application package.

From the lecturer's point of view this (and indeed any other) system, can be quickly adapted and used without much effort (ie. program development). The database approach to the case study produces a much greater understanding on the part of the student, of the system design and what lies behind it. It is also worth emphasising that the lecturer can quickly produce any such working system in dBASE II to illustrate any simple case study idea.

### 5.8.3. The Estate Agents System

This is a case study which can be taken to varying levels of complexity, ranging from a simple file design exercise, to the implementation of a working system. It is essentially a system designed to meet the requirements of a potential house purchaser with the properties available / brought in at the agency. The situation is familiar to most people, and can be easily researched by the student, who can make a casual call on any estate agency in order to view the current manual system. The case study is so popular because of the ability to actually analyze the current system, rather than have the lecturer outline the system to the group, or produce documented evidence of the analysis in the form of memo, company documentation, etc.

#### 5.8.3.1. Database Implementation.

It would be quite possible to end the case study assignment at the design phase, but the availability of the database system means that

the students, having defined their files, can now create them and enter some data. This gives the students a much greater appreciation of what they have actually achieved. Comments here would reflect those in the previous example of the Telecom System. The Estate Agents system has always proved too difficult for any lecturer to produce (in a busy day) a programmed solution, for example in BASIC, although the case study has been used by students to produce a working COBOL system. The use of dBASE II creates the opportunity for the students to quickly implement their file design, and create a simple system with little time spent on implementation. Here the emphasis is where it should be - on design.

The matching processing could become fairly complex, but a simple matching process, based perhaps on price and number of bedrooms, could easily be generated by way of an example. The command file for this can be supplied by the lecturer with little time overhead.

This has been used successfully to introduce programming to Business Studies Students on a BTEC Business and Management Diploma course in Information Processing (IP2) which requires experience in programming. Previously a brief introduction to BASIC had been given, with unsatisfactory results. The group of students subjected to the dBASE version found the package difficult to feel comfortable with, but produced working simple programs (to match requirements, and to produce exception reports) in around 6 hours of teaching.

The use of this case study showed that dBASE II was successful when used at an introductory level in a computing course involving program appreciation.

### 5.8.4. Broadsheet System - Outline of the Case Study

This is the most complex of the case studies discussed. The case study shows how dBASEII can be used to deal with complex systems, and is described in detail in appendix C. The case study begins by introducing the students to a wordprocessor system (WORDSTAR) Together

with some introductory lectures. The students would then work through the case study.

The case study is designed to take the student through a data processing problem, initially involving a wordprocessed solution but developing into an integrated database and wordprocessor system. During the course of the development of the system the student will be introduced to various data processing concepts. The case study is designed around a situation which is easily assimilated by the students involved. The material is complex, and develops into a complex model.

The system deals with the coordination of a team of lecturers teaching a Computing course, but could equally apply to any administrative team. The main aims of the system are threefold:

To keep all team members informed by issuing memos.
To issue meeting details (and agendas) to all team members.
To keep an automated marking system (the Broadsheet).

There are six phases in the development of the system, and it presupposes that the students are familiar with the WORDSTAR wordprocessing package. It is envisaged that the students are given hands on wordprocessing experience at the very beginning of the course, and that this would be their introduction to the course. During the case study the student will be introduced to the database package 'dBASE II'. The seven phases are as follows:

Phase 1 - Develop a simple 'mailshot' memo system to keep lecturers informed. (Utilizing WORDSTAR and MAILMERGE).

Phase 2 - Use the dBASE II package to keep the required input for the MAILMERGE system. This is simply a list of sessions and lecturers, together with lists of students.

Phase 3 - The files created will have 'Update' problems, and in this phase the student will use Entity Relationships and Normalization to design an efficient database system for the Broadsheet. This equates to a simplified Systems Analysis.

Phase 4 - Before implementing the system (using data supplied) a plan and time estimates are produced.

Phase 5 and 6 - A simple implementation using dBASE II with single line commands. The required files will be created and maintained. The student marks will be entered using the database maintenance system. The report generators will be used for all system output. Before attempting this phase the student will have completed an introduction to databases and also the dBASE Educational package.

Phase 7 - This will be a brief examination of the effects of using a menu driven command file system to interface with the user, and the advantages and disadvantages involved.

This case study represents the highest level of system for which dBASEII can be a useful tool. The system is complex enough to require difficult command files, and these are also outlined in appendix C.

In phase one, perhaps as important as developing the skill of generating the memo system using WORDSTAR, the student has an introduction to the idea of commands, and command lines collected into a file. In fact the student has an introduction to PROGRAMMING, but using a wordprocessor. This is seen as a very important part of the case study since the concept of a program has been introduced without the student realizing it (unlike the usual circumstance when the student is only too well aware of it).

The decision constraints that appear in the case study illustrate the kind of evaluations that the students are required to make, for example :

> Compilation of reports is a lengthy process, prone to error, and reminiscent of School Report systems. If the marks were contained in a computer database the reports with marks could be prepared accurately and quickly.

Essentially the task is to design and implement the required computer based broadsheet system. The decisions taken so far reflect that a feasibility study has been undertaken (by the Head of the Computing Section) and the system is deemed to be straightforward to implement. Source data (in unormalized form) looks like this :

Course Title : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Coordinator : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

| Student Names | Session<br>Module 1<br>Lecturer XXXXXXXXXXX<br>M1 M2 M3 M4 M5 EX | Session<br>Module 3<br>Lecturer XXXXXXXXXXX<br>M1 M2 M3 M4 M5 EX |
|---|---|---|
| XXXXXXXXXXXXX | 99 99 99 99 99 99 | 99 99 99 99 99 99 |
| XXXXXXXXXXXXX | 99 99 99 99 99 99 | 99 99 99 99 99 99 |
| XXXXXXXXXXXXX | 99 99 99 99 99 99 | 99 99 99 99 99 99 |

**Fig. 5-1: Manual Broadsheet System Example**

There are, of course, a selection of sessions according to the course.

Before such a system can be implemented, it is important that the user understands exactly what they are trying to do, and also creates the most efficient system. Often such systems are created using the intuitive knowledge of the user. This is simply because a proper analysis takes a good deal of time. Systems developed in this way are

often poor and usually don't work as expected. To develop efficient, reliable and easily maintained systems it is important to use a structured systems development approach. There are many to choose from, but most employ two elements - Entity Relationship determination, and Normalization. Both of these structured techniques produce an efficient file design which can be directly translated to a package like dBASE II.

The student will implement the main files and create some sample data by using the data files supplied, then decide what can be achieved simply by using dBASE II commands, including creating and maintaining all files and entering marks into the appropriate fields. Some sample reports and end-of-session reports will also be produced.

Finally the student is presented with a menu driven system, like the one they have designed, but based around command files. The students can then discuss the advantages and disadvantages of such an approach. Because of the restrictive nature of dBASE II in some areas, it would be necessary to to make some changes to the originally specified system to accommodate dBASE II, which is rather unfortunate. The main problems encountered were:

Only being able to access two files at once.
This led to some complex 'array' handling with memory variables.
(dBASEII has no non-tortuous array handling).

The poor effectiveness of the BROWSE command (otherwise excellent) when handling more than a screenfull of records and the lack of a 'FOR' facility with it. This had dire effects in that the marksheet file had to be split into smaller files, one for each session - so that they fit into one screen. This could be achieved with command files for creation and updating.

The student will see only the working system on disc. For an overview

of the system look at the two menu programs called MENU and REPS.

### 5.8.5. Summary

This final case study illustrates a simple fact. It would be unwise to embark on the use of dBASEII with a group of students, for other than reasonably simple tasks requiring flat files, or at most two tables. Relational systems cannot be properly created in dBASEII because of the two file restriction, and this caused a great deal of complexity in programming what would otherwise have been a simple system. In the context of the group of students targetted, the aim was to demonstrate to them the final system, and not to teach them how to generate the final system themselves.

It is doubtful if this case study would prove successful when used with totally naive or casual users. The most successful part of the case study concerns the first phases which deal with mailshot memos. The exposure of the students to the material does give them a clearer understanding of the kind of work involved in developing such a system, if not a clear understanding of how the development is implemented.

The clear implication is that dBASE II is not suitable for the development of complex systems for educational aims.

The value of using the database system is that any design errors on the part of the student are quite easy to rectify in most cases. Once any data has been input, it can easily be manipulated within the database to suit the required design needs. The students all worked on a team basis.

There is an article which relates to some of the case study concepts discussed here. The article (5.8. MacCallum Stewart L 1985) is about teaching programming as a team effort, and many of the learning advantages outlined in the article will apply to the case studies outlined above. In particular the team case studies would form a good

basis for the kind of programming work discussed in the article.

In practice the students learn a lot about working as a team, the problems of getting correct data inputs, the amount of time needed for the various stages in the project, as well as the more normal 'computer skills' such as knowledge of file types, indexes, relational design and so forth. Experience at Basingstoke has shown that the students enjoy the case study and that it will form a valuable core experience which can be used in various modules later in the course. The students had gained valuable practical knowledge and skills in many areas which are dealt with on a theoretical basis later in the course. The lecturers noticed an increased awareness in the students, and an ability to discuss more expertly the questions related to the case study which arose later in the course modules.

These kind of skills have proved very useful when the students come to design and implement systems in COBOL, and this is discussed later in the thesis.

As part of a 'holistic' approach a lot can be achieved through the use of case studies. The availability of dBASEII means that the lecturer can adopt a flexible approach to an implementation which will stress the design aspects of any system. The next section discusses the CAL package which comes with the dBASE Educational system.

### 5.9. CAL Packages

It seems strange that the computing curriculum area makes relatively little use of CAL packages. The only major CAL package I have seen used in a school or college specifically for a computing course is dBASE Educational.

This section describes the CAL package that can be bought on disc and is called 'dBASE II Educational'(Ashton Tate Release 2.0 1984). The package is rather interesting, and most closely resembles the 'learn' utility on the unix system (although it is not quite so clever in that

the user cannot interact with the system and then return to the package, everything the user does is monitored by the package before being passed to dBASE to implement).

Other CAL packages could be written, on the same principal, in order to teach any aspect of data processing using dBASE II facilities.

Completely computer based, the CAL package has no manuals or associated materials except a supply of three Ashton Tate related text books which deal with relational systems in general but do not relate to the educational package. It consists of an introduction to relational data bases, to the CP/M system, and to dBASE II in particular. Detailed below is the introduction from the package:

'There is something unusual about these lessons, there is no manual. In keeping with the computer age, we have chosen to put information on magnetic disk, rather than on paper. We think that working interactively with a computer is a far better way to learn than by simply reading a manual.'

In this course, you will be actively working with dBASE II. In many instances, you will decide the data to be used and how it will be displayed or printed. Where data is supplied, you will have to issue the proper commands to act on the data. Don't worry though, the dBASE II Lessons give you a lot of help when you make errors, so if you make errors in dBASE II, you'll know how to correct them. '

'For the beginner, we believe you will find that the first four lessons will be all you need to really feel comfortable with dBASE II, and at that point, you will know all of the operations required to create files, generate reports, change data on your files through editing, and perform computations on your data. It will probably not take you more than three or four hours to get through these first four lessons.'

### 5.9.1. Lesson Content

This is a list of the general contents of each of the ten lessons, and this gives a good impression of how the lessons build up the students knowledge of dBASEII and also related concepts.

Lesson 1

- Creating a File
- Display The File Structure
- Append Data Into The File
- Displaying the Records
- Generating Report Forms
- Report Generation
- Leaving the dBASE II environment
- Lesson Summary

Lesson 2

- USEing TEST
- STRUCTURE of TEST
- Ammending Records
- REPLACE command
- GOTO Command
- DELETING Records
- PACKing the data
- RECALLing deletions
- SKIPing records

Lesson 3

- SCOPE and FOR introduction
- NEXT scope
- ALL scope
- FOR conditional
- Conditional Operators
- Computations
- Computing with REPLACE
- SCOPE's Available

Lesson 4

- Ordering Records
- INDEXing Records
- FINDing Records
- When to use INDEX
- LOCATE and CONTINUE
- LOCATE ALL FOR
- BOOLEAN Operators

Lesson 5

- File Management
- DISPLAY FILE
- COPY TO for backup
- Command Files
- DO Command File
- DISPLAY FILES LIKE
- DELETE FILE
- MODIFY STRUTURE
- APPEND FROM

Lesson 6

- SORTING
- UPDATING a file
- Memory Variable Commands
- STORE TO
- DISPLAY MEMORY
- SAVE and RESTORE
- SAVE TO
- RELEASE TO
- RELEASE ALL
- RESTORE FROM

Lesson 7

- JOIN
- PRIMARY and SECONDARY

Lesson 8

- Functions
- ACCEPT
- The @ Function
- The $ Function
- The ? Function
- Keyword Searching
- The ! Function
- The ?? Command
- Using other Functions

Lesson 9 (Screen Formatting)

- Formatting with @

Lesson 10

- Command Files
- Control and Macros
- Example File
- DO Command

### 5.9.2. Summary

These CAL dBASE II lessons have been reasonably well designed, and are certainly useable under guidance. Most students tend to get 'stuck' at various points in the series, where the lessons have failed to take account of the novice user, and it would seem that the lessons have originally been either not tested, tested on programmer users, or tested on highly intuitive novice users.

As an example, in lesson 1 (and this causes a lot of problems because it occurs early on and users lose confidence) the description of setting up fields for a file (inside the create command) and for entering data are overlayed in a confusing way. Most users ask for guidence here, and those who don't generally enter data instead of the field names at this point.

The introductory section, is also confusing for novice users. There is a lengthy discussion in it of operating systems and some commands. The description of the PIP command for CP/M users is incomprehensible to most novices. This causes problems not just in itself but because this is the basic introduction to the system.

However, with some expenditure of time the lessons could be tailored to any particular school or college needs. All of the lessons are themselves dBASE II command files which could be changed. The main problem in this respect is that the files use a complex system of macro calls and are totally undocumented, and this would mean that any maintenance would require a considerable input of time simply in understanding / documenting the current system. The normal solution is to lecture the students around the problem areas in the CAL package. It is important to realise that the later versions of dBASEII and especially dBASE III have comprehensive on-line help and tuition facilities which to some extent replace the CAL lessons described here. It may be worth waiting for Ashton Tate to provide their own more sophisticated systems solutions to the novice user problem, which can then be used in an educational context.

The three main user problems arise then, due to :

1. The introduction begins with a lengthy explanation of the CP/M PIP command, which is totally incomprehensible to most people (the PIP command that is) and in the context of the novice user is a total dead loss at this stage.

2. The inclusion, in lesson one, of the syntax error correction system (which is very confusing even for experienced users) proves totally confusing to novice users. Whilst this may be considered to be a flaw in the dBASE II system itself, it could easily be either missed out, or put in to a separate teaching session.

3. The way in which the definition of fields is introduced (in the lesson one session) will confuse many users (around 50%) who attempt to enter data instead of define field names. This arises because the lesson attempts to cram in too much information (for example about numeric fields) which isn't then used in the session. The lesson fails to properly explain the concept of defining field names and types.

However, it must be said that all of the lessons are accessible (in the form of the dBASE Command Files), and could be changed by an enthusiastic user.

It should also be said that other sets of CAL lessons could also be created quite easily (although this would be time consuming) for all aspects of a computing course.

In general it would be unwise to use the CAL package on its own, and it is best used after an introductory lecture as a means of reinforcing and adding to the lecture content. If fully developed the CAL package could be a highly useful novice learning aid. The newer ASSISST system provided with dBASE III is far superior in this respect. The possibility of developing useful CAL packages using the system is a potentially very powerful one. It is perhaps a reflection of the current state of educational software, that so much time and effort have been devoted to duff BASIC software, which may so much more successfully have been devoted to producing viable teaching packages on good 'learn' systems like dBASE Educational.

### 5.10. CAL Example Modifications

Modifications have been made, by the author, to the lessons with the three problems outlined previously. Details of these changes are given in appendix B.

#### 5.10.1. The Lesson Files

The CAL package consists of a series of files, which run over dBASE II, dividing into command files (.PRG endings in MSDOS), database files (.DBF extensions), associated index files (.IDX), report form files (.FRM) and a memory variable file (.MEM). The files are listed below.

### 5.10.1.1. Command Files

These are the command files involved in the CAL system:

| | | | | | |
|---|---|---|---|---|---|
| LESSONS | PRG | 4608 | TEACH4L | PRG | 3072 |
| TEACH1 | PRG | 10240 | TEACH4 | PRG | 7168 |
| TEACH1R | PRG | 5120 | TEACH4B | PRG | 2048 |
| TEACH1S | PRG | 3072 | TEACH5 | PRG | 11776 |
| TEACH11 | PRG | 5120 | WHITEHSE | PRG | 2048 |
| TEACH1R1 | PRG | 9216 | TEACH6 | PRG | 5632 |
| CHECKC | PRG | 2560 | TEACH6U | PRG | 1536 |
| CHECKC1 | PRG | 2048 | TEACH6M | PRG | 5632 |
| TEACHXC | PRG | 1024 | TEACH7 | PRG | 7680 |
| TEACHI | PRG | 8192 | TEACH7P | PRG | 5632 |
| TEACH21 | PRG | 3584 | TEACH8R | PRG | 2048 |
| TEACH2 | PRG | 6144 | TEACH8 | PRG | 7168 |
| TEACH2R | PRG | 1536 | TEACH8F | PRG | 5120 |
| TEACHC | PRG | 1536 | CLEAR@ | PRG | 2048 |
| TEACH3 | PRG | 5120 | CHECKS | PRG | 2048 |
| TEACH3F | PRG | 4608 | CLIENTLS | PRG | 3072 |
| TEACH3C | PRG | 4608 | TEACH10 | PRG | 4608 |
| TEACH3F1 | PRG | 512 | TEACH9 | PRG | 7168 |

36 File(s)

**Fig. 5-2: CAL Package Files**

### 5.10.1.2. Database Files

| | | | | | |
|---|---|---|---|---|---|
| T1RDBF | DBF | 1536 | JOBHIST | DBF | 1024 |
| MASTE | DBF | 1024 | TAXMAST | DBF | 1024 |
| HELLO | DBF | 1536 | EMPNAME | DBF | 1024 |

```
EMPMAST  DBF      1024
EMPTRANS DBF      1024
TUTNABK  DBF      1536
COMPUTE  DBF      1024
EMPBAK   DBF      1024
COMPUTE1 DBF      1024
EMPTBAK  DBF      1024
EMP8     DBF      1024
NAMEADDR DBF      1024
DEPTMAST DBF      1024
EMPMSRT  DBF      1024
EMPTSRT  DBF      1024
JOINJOB  DBF      1536
JOINPAY1 DBF      1024
JOINPAY2 DBF      1536
JOINPAY3 DBF      1536
EMPBAK1  DBF      1024

       25 File(s)
```

### 5.10.1.3. Report Form Files

```
COMPUTE  FRM       512
EMPMASTR FRM       512
EMPNAME  FRM       512
JOBHFRM  FRM       512
JJFRM    FRM       512
TAXMAST  FRM       512
DEPTMAST FRM       512
EMPMAST  FRM       512
REGFRM   FRM      1536
EMPMFRM  FRM      1535
       10 File(s)
```

### 5.10.1.4. Index Files

```
EMPNOIND NDX      1024
NAMEIND  NDX      1024
       2 File(s)
```

### 5.10.1.5. Memory Variable File

```
TUTMEM   MEM      1024
       1 File(s)
```

#### 5.10.1.6. The Operating System Modification

In order to modify the references to the operating system lesson content, file TEACHI.PRG was modified as shown in appendix B. This replaced the deleted section on operating system details in the CAL package. The intention was to remove the details of operating system functions from the package.

#### 5.10.1.7. Syntax Modification

This was quite easily changed by deleting the section from the file TEACH1S.PRG. The difficult part was identifying the appropriate file to change. However, the students do have to be taught about the system. In general the best method here seems to be a verbal discussion. The system does not seem amenable to CAL implementation because of the complex nature of the syntax changing system.

#### 5.10.1.8. Field Definition Modification

Again, the file concerned is TEACH1.PRG. Essentially the task was one of changing the text messages to make them more understandable to the novice student. In essence the text gives more direction on two points:

This is a field definition phase and no data is entered
Data definitions are for a name, address and phone number
file and will be character information

The altered portion of the TEACH1.prg file is listed in appendix B as it appears in the dBASE II system.

#### 5.10.1.9. Modifications Summary

The modifications took some time, and have not yet been fully tested. In general, provided the changes relate only to the text displayed,

and do not affect the logic of the program, then any changes are straightforward, but time consuming, and potentially the cause of errors in use.

## 5.11. Summary

This chapter has discussed, through examples, the way in which the associated DBMS tools can be used by the computing teacher as the focus for the core part of the computing course. It should be clear that many course objectives can be met simply by careful use of these tools. The integrated use of these tools can form the core of a computing course. The thesis assertion is that the use of a DBMS can generate the core part of a computing course, together with the other tools which also need to be used.

This contrasts strongly with a course based on the language interpreter BASIC, commony used especially in schools as the main tool for learning. The students using microcomputers mainly as machines for learning programming are receiving a very narrow view of the computing course.

One aspect of the computing course, programming, is worthy of special consideration because of the procedural language of dBASEII.

dBASEII uses a procedural language to allow the user to build systems and perform all of the tasks which cannot be done using the available single commands. This approach is not the 'standard' one, when industry seems to be adopting SQL as the standard approach, for example SQL will be the database option on the new IBM personal system range of microcomputers. It is the procedural language of dBASEII that makes it so versatile from the point of view of the educational user, because it spans so many aspects of the computing course experience.

When the educational user experiences dBASEII they are not just experiencing a database system, they are also going to gain experience

in using a programming language.

The next chapter discusses some aspects of using dBASEII as an introductory tool in teaching programming concepts.

## 6. Chapter Six - Programming

The programming language is another tool which can be used in a computing course. There has been a long history of criticism aimed at the teaching of computer programming. Many teachers consider the learning of the necessary skills to be a mysterious process which many of their students fail to follow successfully. The usual approach is a bottom up (you have to learn how to do everything before you can learn to do anything). However, at the end of this process the students are supposed to acquire a top down methodology for program design. It is little wonder that, faced with this approach, many students fail to become good programmers.

The author has used BASIC, COMAL, COBOL, SIMPLE (on ABS minicomputers), and dBASEII to teach novice users about programming. SIMPLE is an interesting minicomputer language, similar to COBOL although maintaining a central data dictionary for the user so that data definitions are seperated from programs. This is similar to the effect of dBASEII, and the author's experience leads him to suggest that novice students learn more about good programming and programming concepts when taught in this environment than when taught using a language.

This thesis is not about teaching programming. This is a complex topic worthy of a thesis in its own right. The thesis does have something to say about introducing students to the concepts of programming. It is the author's belief that a programming language is not the only or neccessarily the best way of introducing students to programming concepts. One of the great advantages of dBASEII (and some other DBMS) is that a procedural programming language is used. In terms of DBMS this may not be a good thing. In educational terms it makes the DBMS very interesting. It should be possible to use this procedural language to introduce students to programming concepts before they use a language interpreter. Experience of this was

discussed in chapter four. This chapter discusses those aspects of dBASEII which make it useful for teaching introductory programming concepts, together with the drawbacks to this approach.

It is not suggested that dBASEII, on its own, can be used to teach programming to computing students.

## 6.1. Background

Structured programming would seem to be a well established philosophy, which can be traced back at least to the writings of Edgar Dijkstra in 1968. How well established is difficult to decide, and it is still common to find articles, like the one entitled 'Why Structured Programming' (6.1. Atherton R,1983). Many teachers have taught themselves programming and the ideas of structured programming are by no means widely understood in schools and colleges.

BASIC is often seen as the culprit, for since it is possible to write carefully structured programs in BASIC it is by no means easy, and there is a great temptation to produce the program at the keyboard. In an article by Boris Allen (6.2.,Allen B,Practical Computing,May 1985) however, he points out that the adoption of PASCAL, being a structured language, is no sure measure of structured program teaching. He states that 'If you take almost any textbook for students of programming in Pascal, the sequence of topics is more or less bottom up, and the succession of topics differs little from those of BASIC texts. Students are discouraged from using BASIC because it promotes a bottom-up attitude, yet those same students are taught in a bottom-up manner, mainly because it is easier for the teacher.'

The argument against bottom-up teaching is that the student needs to know a good deal, particularly about mathematical type concepts, in order to progress. This is, perhaps, why so much emphasis is put on a good mathematical background by so many teachers. It is, of course

quite unnecessary, and using a top down approach the required concepts can be taught as and when they are encountered, and after the student has become proficient at the higher level.

The advent of LOGO with the turtle could change the way in which programming is taught in schools, as will Micro-PROLOG. Students will write interesting programs straight away. By using procedures the students will learn about arithmetic, data structures, and so forth as they are required. Ken Bowle's book (6.3,Bowles K,1977) links Pascal with a graphics turtle, and Bowles reports that when using the top down approach mathematically inadequate students are generally as adept in learning to write programs and solve problems on the computer as are the students that arrive with a stronger mathematical background.

If students are introduced to programming through a database management system then the top down approach is forced on them, there is simply no other way. There is no requirement on the part of the student for any particular high level mathematical concepts, and in general the programming required is simply common sense - and certainly requires little mathematical conceptual knowledge. In short, the environment of programming within the database system makes an ideal starting point for the top down methodology which needs to be developed later.

There is only one real constraint, which is that the database system will emphasize the file based commercial aspect of programming, rather than the scientific emphasis. Even so, the experience will be of benefit in any later programming course.

It is not the main purpose of the thesis to look in detail at how to teach programming, and database management systems like dBASE II are far from ideal environments if the student spends a lot of programming time with them, however as an introductory medium they seem ideal.

## 6.2. RDBMS Programming Features

In order to program within the environment of a RDBMS eight features are required:

Selection (rows)
Projection (columns)
Join (tables together)
Intersect
Union (add to data)
Subtract (remove data)
Product
Division

These features were established by Codd and led to the development of his RDBMS language DSLAlpha. It does not necessarily follow that all of these requirements are present in implemented systems. Few will be found in dBASEII which opts instead for a utilitarian approach to what the user will find immediately useful and easy to understand. Features in dBASEII are geared toward the kind of features normally sought by a traditional COBOL programmer. dBASEII uses a procedural language which in many ways is similar to BASIC. This is why it is possible to think of using dBASEII to teach programming concepts. A student using command files in dBASEII will be learning how to write programs.

Essentially the use of dBASEII will allow a trainee programmer to gain a good introductory grasp of many important programming skills, including structured programming techniques and files. The student will be free to concentrate on these aspects, rather than the lower level considerations of programming techniques enforced on them in systems such as BASIC. This is not to suggest that such skills are not needed, although with the advent of the widespread use of fourth

generation languages this may prove to be so. Rather, the student will find the skills learnt a useful introduction to programming skills.

In particular the student will find it difficult not to adhere to structured top-down techniques. The traditional programming student tends to produce confused and obscure program logic using bottom-up techniques, which BASIC seems to encourage. Willmott (6.4,Willmott G M R,1983) states "Recently a great deal of thought has been given to methods which will ensure the production of accurate programs in a given time scale with the same predictability as can be given to other production techniques. This in turn has led to the creation of languages such as PASCAL and COMAL which offer the programmer precise facilities with which to express his requirements. Such languages offer competition to the predominence of BASIC whose disadvantage lies in the fact that it permits the haphazard and unstructured writing of programs. Such programs are difficult to test, understand and maintain".

## 6.3. File Design

This is a key part of producing specifications for any commercial system, and even if the system required is very simple (for example a telephone directory) then some aspect of file design is necessary. Structured analysis tools like Data Flow Diagrams, Entity Relations, and Normalization are also recommended. An understanding of sequential and random access files (perhaps especially ISAM files) is learnt through experience at the very beginning of the exposure to the database package, as are the concepts of key field indexes. In a normal bottom-up approach these are often the last concepts to be reached (if at all).

## 6.4. Programming Environment

One of the great benefits of BASIC is its apparent ease of use for a first time user. However, many BASIC systems are not in fact as good as they might appear.

Basic environments differ a lot from system to system, and the older style line editors are gradually disappearing. In these older editors it was necessary for the student to learn a fairly complex set of control key commands in order to edit lines, although the lines could be re-typed from scratch. Many of the effects of the control keys were transparent to the user, and so student's found this a difficult system to use.

Newer environments, and the BBC B is a good example, employ some kind of screen editing, although in the BBC B this is a curious combination of screen and line editing, which is nonetheless easy to use.

With interpreted BASICs the environment is generally straightforward, but it must be said that the overall environment leaves a lot to be desired. The BBC B for example begins with this display (the BASIC is in ROM):

```
BBC MODEL B

ACORN DFS

BASIC

>
```

Inevitably, the novice user wants to know what it means, so before anything else is achieved a lengthy and usually unsuccessful attempt is made to explain the DFS message. In other words the screen displays which experienced users take for granted are often quite confusing for beginners, and this is particularly so for mature

students who are not at ease with displays which they don't understand.

Error messages are usually poor as well. Later versions commonly found in schools and colleges would include RMBASIC which is a great improvement, with automatic indenting of program lines to show structures, for example.

To any BASIC user of five years ago, the COMAL environment was a real eye opener. Although COMAL is now a rather old concept (around five years in Britain in common use), and may not now seem so spectacular, in context of five years ago COMAL seemed quite revolutionary to BASIC users. The editing is achieved by the usual EDIT linenumber type command, but then the line is displayed, and the user makes all corrections with just four keys - move left, move right, open text, and close text. These are specially marked keys. Every novice user can understand and use this system immediately.

All program structures are force indented in all program listing, and so must be closed correctly, and even better, the syntax is checked line by line as it is typed in and so students do not spend lengthy periods keying in lots and lots of syntax errors which only emerge on running.

In all the environment is so pleasant and simple to use that a conventional BASIC system looks poor beside it. Again this reflect the care and thought that went into the novice and educational aspects of this implementation. The author has used COMAL with schoolchildren following CSE and O level courses and found it to be much more successful than BASIC, or indeed any other system experienced.

### 6.4.1. dBASE II Environment

The environment is not good, but is no worse than the normal BASIC

environment. There are problems with error messages, which are not generally helpful, and novid users find correcting mistakes particularly difficult. The environment is far from perfect for novice users, and reflects the fact that the choice of dBASE II is not an ideal one. The correction environment is discussed in the section below.

#### 6.4.1.1. Correcting Mistakes

This is the area in to which many novice users fall unawares, and which presents the most problems for them.

When an error is made entering a command dBASE II will display an error message similar to the one below (this is similar to the example used in the CAL package dBASE Educational).

```
*** SYNTAX ERROR ***
         ?
      command

CORRECT AND RETRY?(Y/N)
```

The user then has the opportunity to correct the command. If, for example the user misspells USE when the command USE NAMES is keyed in. Then the following dialogue might ensue (user inputs are highlighted).

```
.UES NAMES
*** SYNTAX ERROR ***
         ?
CORRECT AND RETRY (Y/N)?:Y

CHANGE FROM:ES
CHANGE TO:SE

USE NAMES
```

MORE CORRECTIONS:N

This is a simple concept, but proves difficult for novice users. An effective system when the command to be changed is long (for example forty characters), but often it's easier to re-enter the entire command, rather than go through the CHANGE FROM, CHANGE TO procedure. To do this, respond N when the CORRECT AND RETRY (Y/N) message is displayed. A period (.) will then be displayed, and the command can be re-entered in it's entirety.

A brief glance will show that this is very complex for the novice user, and definitely the cause of most first time user complaints. It would be preferable to make the novice user key in the commands again, but the error routine is automatically invoked by the system. If the user attempts to key in changes but types the wrong thing then they are locked into the routine and will spend a long time getting out. Often, by the time they do extricate themselves they are so confused that they don't realize they are back to the system dot prompt.

## 6.5. Design Consideration

Teaching programming should be very much about teaching design. One of the great features of dBASE II in this respect is that the commands are at a high level then they are, in essence, psuedo code, and if the student is taught to use the dBASE commands, then they are also taught psuedo code. Some form of formal structured system should be taught, and this is a recommended plan for a program development cycle:

Review Specifications
IPO Chart
Psuedo code, and/or Structure Diagrams
Design Review (Peer Review)
Coding
Code Review (Peer Review)

Key in,test, and debug
Document

**Fig. 6-1: Program Development Cycle**

Structure Diagrams (see for example 6.5,Peltu M,1983) or 'Structured System Design Diagrams' as they are sometimes called, have proved very useful in constraining students to top down structured methodologies. These are echoed in JSP or Jackson Structured Programming, which is now a standard system in many colleges.

The adoption of this cycle in the school classroom, with students organized in teams would have two major effects:

Correctly designed programs would be implemented
Students would learn to work as a team and hardware resource demands would be lower.

The same design cycle and techniques are transferable to most language systems.

## 6.6. dBASE II - The Language

Before the usefulness of dBASE II in particular is discussed, it is necessary to look at the language it offers. This overview compares the language of dBASE II with that commonly expected by a BASIC programmer. A COBOL programmer would more readily accept the validity of the dBASEII language since many of the features enhance those available in that language. There are, as discussed, limitations, especially with regard to the constraint of having only two files open at once. By describing the facilities available in dBASE II it should be apparent that an environment similar to BASIC is available, but the environment offers much more scope.

This is the introduction to the language offered by Ashton Tate (6.6 .,Jenner S C,1982):

"For the experienced programmer, you're in for a treat. If you think COBOL, PL/I, FORTRAN, PASCAL, or BASIC are nice programming languages, wait till you try this. The combination of structured English, set processing, dBASE II functions, macros (substitutions within commands), stored command procedures (programs), nested DO and nested IF capability, along with true relational database capability is extremely powerful. At the same time, dBASE II is very easy to use. Since all of this is available on a microcomputer makes it even more attractive."

The discussion is divided into four sections; functions, structures, files and the higher level facilities offered by the database package. The notes follow those in the on-line help facility in dBASEII, indicating the potential benefit to a novice student of this help system.

### 6.6.1. dBASE II Functions.

Many of these are familiar to BASIC users, and the following represent a selection.

CHR -- CHR(<numeric expression>) - yields the ASCII character equivalent of the <numeric expression>. e.g. ? CHR(7) rings bell

DATE() -- returns the character string that contains the System Date in format xx/xx/xx.

EOF -- end-of-file function evaluates as True if an attempt has been made to go past the last record in a database.

FILE -- FILE(<file>) - existence function evaluates as a logical True if <file> exists on the default drive, and as a logical False if it does not.

LEN -- LEN(<cstring>) -- length function returns the number of characters in <cstring>. ? LEN('HELLO')

TRIM -- TRIM(<cstring>) - trim function removes trailing blanks from <cstring>. ? TRIM('HELLO ')+' THERE'

### 6.6.2. dBASE Language Structures

Whilst, as with BASIC and COBOL it is possible to write unstructured code in dBASE II, all of the required structures are present, and in addition the CASE structure is available.

The DO LOOP is the only loop structure and equates to a while - endwhile structure. A counted loop would have to be set up within the do loop.

The DO LOOP

```
DO WHILE <exp>    --  used in command files to open a structured loop.
   <commands>         Commands in between are executed so long as the
[LOOP]                DO WHILE <exp> is found to be True.
   <commands>
 ENDDO
```

```
e.g.  USE MAILLIST
      DO WHILE .NOT. EOF
          ? NAME
          ? PHONE
          SKIP
      ENDDO
```

There is a properly structured IF statement of the form:

The IF - ELSE - ENDIF Structure

```
IF <exp>                -- in command file only
   <any statements>     ELSE clause is optional.
 [ELSE
   <any statements>]              e.g., IF STATE = 'CA'
 ENDIF                                     DO INSTATE (command file)
                                     ELSE
                                         DO OUTSTATE (command file)
                                     ENDIF
```

This is much better than usually offered by BASIC, and is the same as offered by COBOL.

The CASE statement is not generally offered by either BASIC or COBOL and represents a definite bonus for dBASE II users.

DO CASE -- used in command file to choose one and only one of several possible execution paths. OTHERWISE clause optional, and executes when no CASE is true. ENDCASE is needed to close command.

```
e.g.    USE MAILLIST
        ACCEPT "WHICH MENU OPTION DO YOU PREFER?" to CHOICE
        DO CASE
              CASE Choice = '1'
                   DO Labels
              CASE Choice = '2'
                   DO Addnames
              CASE Choice = '3'
                   DO Edit
              OTHERWISE
                   QUIT
```

ENDCASE

### 6.6.3. dBASEII - File Handling

In order to create a file the user simply types the Create command, and the system will prompt the user for the file structure:

CREATE [<filename>] — Creates a new database file.

Once a database has been created, data can be entered by using the APPEND command:

APPEND — allows user to add new records to database in use. (If index is also in use, the index file is automatically updated).

Obviously this is a very powerful command, and this means that a user can key in data to the file immediately after the CREATE command is completed. In a similar fashion the EDIT command can be used to change the contents of any record in the file.

EDIT [<record number>] — enables selective editing of database in use by record number. Requests record number if not supplied. When edit of particular record has been completed, Ctr-W brings back EDIT's record number prompt (ENTER RECORD £:). To terminate EDIT mode, answer record number query with a return.

If the user wishes to locate a particular record in a sequential file

LOCATE [<scope>] FOR <exp> — finds first record of database for which FOR <exp> is True. Use CONTINUE to find next such record. (User may manipulate record before resuming search with CONTINUE.)

```
e.g., . LOCATE ALL FOR ZIP >= '95000' .AND. ZIP < '96000'
       RECORD: 00123
     . DISPLAY
     . CONTINUE
   RECORD: 00232
```

Of course, sorting the file might be a good idea first, so :

SORT ON <field> TO <file> [ASCENDING / descending] -- writes a new copy of the database in use with all records arranged in order. Uses ASCII value to determine the order (generally Spaces, Numbers, Uppercase, Lowercase, then Symbols). SORT will not copy records marked for deletion. Default order is ASCENDING.

```
e.g., . USE MAILLIST
      . SORT ON ZIP TO MAILZIP DESCENDING
```

Indeed, dBASE II offers a complete batch processing environment:

UPDATE -- allows batch update of pre-sorted or indexed database by drawing information FROM designated database (pre-sorted on same <key>). Keys of records in USE and FROM databases are compared for match. dBASE can then ADD the numeric fields of the FROM database to corresponding fields in the USE database. It can also REPLACE character or numeric fields of USE database with the contents of corresponding fields of the FROM file.

Syntax: UPDATE FROM <file> ON <key > [ADD <field list>]
[REPLACE <field list> or <field> WITH <field list>]

When files are first created they are sequential files, but indexes

can later be created very simply:

INDEX ON <cstring> TO <index filename> -- creates an index file for database in use based upon designated index 'key', i.e., the <cstring>. Usually index will be 'keyed' on a field name.

This is equivalent to ISAM file handling, and this facility is rarely offered in BASIC and certainly not in the varieties available in schools - even on the machines which will run dBASE II. The ISAM facility makes dBASE II fundamentaly better than BASIC. Having created an index the user can FIND a record quite simply:

FIND <cstring> -- when using indexed files, positions to first record indexed by <cstring>.

The fact that using indexed files is so simple in dBASE II means that all users will understand the concepts quite quickly.

### 6.6.4. dBASE II - Database Handling

Some of the preceding commands are very powerful, but dBASE II offers many more commands which are more specific to a database system. The BROWSE command :

BROWSE [FIELDS <field list>] -- Brings up Full-screen viewing and editing of the database in use.

allows the user full screen editing of the database (file) in use through a window which can be scrolled up or down through the records in the file. More comprehensive data changes can be made with the CHANGE command :

CHANGE -- Permits Non-Full-Screen editing of database in use by field. Hit ESCape key to terminate CHANGE mode.

Syntax: CHANGE [<scope>] FIELD <list> [FOR <exp>]

. CHANGE ALL FIELD ZIP FOR ZIP = '90045'
RECORD: 00123
ZIP: 90045
CHANGE?
(Enter the characters to be changed and hit return. Add new data at the TO prompt, or hit return to go to the next appropriate record.)

A similar command, REPLACE works this way:

REPLACE -- allows user to replace contents of specified fields of database in use. If index file is keyed on field targeted for update, index in use will be automatically updated. Default <scope> is current record.

Syntax: REPLACE [<scope>] <field> WITH <exp> [,<field2> WITH <exp2>] [FOR <exp>]

e.g., . USE MXPROJ
. REPLACE ALL COST WITH COST*1.1 FOR ITEM = 'ELECTRIC'

There are several commands for producing file statistics:

SUM -- computes and displays the sums of numeric field(s) of database in use. The <scope> option permits selection of the range of records to sum: FOR <exp> allows summation on particular criteria. TO <memvar> stores sums to the designated memory variables. Default value of <scope> is ALL non-deleted records.

Syntax: SUM <field> [,<field2>] [<scope>]
[TO <memvar list>] [FOR <exp>]

e.g.,

. USE SHOPLIST [fields are ITEM, NUMBER purchased, COST of item]
. SUM COST * NUMBER FOR ITEM = 'food'
. SUM NUMBER FOR ITEM = 'hardware' TO HARD
. SUM NUMBER, NUMBER * COST FOR ITEM = 'hardware' .AND. COST > 9.00

TOTAL — creates a summary version of an indexed or pre-sorted database by copying only records with a unique <key>. Specified <key> must be key to the index or the key upon which database is already sorted.) Records with duplicate keys are removed. All records with the same <key> can have their numeric fields totalled in the TO database by using the FIELDS option.

Syntax: TOTAL TO <file> ON <key > [FIELDS <field list>]

dBASE II has its own report generator :

REPORT — used for creating a Report Form file (FRM) for displaying specified information from a database in a user-defined format. Outputs results to screen or printed page.

Syntax: REPORT [FORM <form file>] [<scope>]
[TO PRINT] [FOR <exp>] [PLAIN]

Finally, the relational JOIN operator is available, and this can be used quite effectively to overcome the two file limitation in many circumstances.

JOIN -- creates a new database by combining the records of files

in use in Primary and Secondary areas. Records are added where FOR <exp> evaluate as TRUE. Command must be executed from primary area. Default on FIELD <list>] to all.

```
e.g., . USE NAMES
      . SELECT SECONDARY
      . USE MAILLIST
      . SELECT PRIMARY
      . JOIN TO NAMEML FOR LAST <> S.LAST
```

### 6.6.5. Command Files

Programs in dBASE II are called command files. All of the preceding commands can be used in immediate (interactive) mode although the structural commands would not work when used in this way, and can also be put into a batch file, together with the structural commands.

This is a fragment of a command file, illustrating the read statement which has an effect very similar to that in CIS COBOL ACCEPT statements.

READ -- used in command files to enter full-screen mode for entry or editing of variables. The Full-screen prompts and window are created by @ SAY commands with GET phrases.

Format file fragment:

```
STORE '                    ' TO name
STORE '              ' TO phone
@ 4,4 SAY 'Name' GET name
@ 6,4 SAY 'Phone' GET phone PICTURE '(@@@)@@@-@@@@'
READ
```

## 6.7. Can dBASE be used ?

From this discussion of the procedural language of dBASEII it is easy to see that in many respects this is a much better teaching tool than BASIC, and has many attractive features for novice programmers. Importantly, all of the required program structures are there, and equally there are several distinct drawbacks discussed later in this chapter.

In order to introduce students to programming, the concepts of structured design, program structures and files should be the most important and also the first concepts introduced. With dBASEII the student can build one or more files of data and be introduced to file concepts before writing any programs. Having created files the student can learn about maintaining the data on the files, again without having to write programs.

By then discussing small programming tasks using these files, which involve one or more of the required program structures, the student can write simple programs (command files) which demonstrate structures in a way that the student can relate to. These programs will be quite short, but will perform realistic tasks on the data files. Students will be able to see realistic data processing output with simple short programs.

Since some of the commands are so powerful, the dBASE code will tend to look like the psuedo code used to develop designs for programs in other languages, for example BASIC or COBOL. The student is dealing with programming the system at a much higher level than would normally be the case, yet in a language style familiar to BASIC or COBOL programmers. The student is taught to think about program design at this high level and so adopts a top down approach more readily when other programming languages are used.

## 6.8. Comparing dBASEII with BASIC

The language facilities offered are similar to those offered by many versions of BASIC (for example the LEN function) in some respects. However, there are many commands which are much more powerful than the BASIC user is used to. The file definitions are stored separately, and so the language overcomes the problems of the non declaration of file fields in BASIC.

The language has a 'familiar' feel to BASIC programmers, and this could be considered important when many computing students have already programmed in BASIC to some extent.

From the preceeding discussion of the nature of the dBASE language it seems clear that for many computing students the aims and objectives of a typical programming course can be met largely, but not completely by dBASEII and that this seems likely to be much more successful than using BASIC, especially in the light of previous experience at Basingstoke.

## 6.9. The Problems

The dBASEII language has many advantages when used for teaching the novice programmer. Equally it has several major drawbacks, among the most notable are:

Parameter passing in procedures

In dBASEII command files can call other command files, effectively this is a procedure facility, but parameters cannot be passed other than as memory variables.

Data typing in memory variables

Memory variables have to be initialised before they are used, for example with : STORE 'HELLO' TO STRING. There is

no way of looking at a memory variable name and getting a system indication of its contents, as can be done in BASIC with $ and % indicators. This means that the student programmer has to be very careful when using memory variables. The best way to use them is to make the student declare all variables at the start of the program, with STORE statements and notes.

Array handling

The dBASE language has no array facility and any attempt to handle arrays is complex, and unsuitable for novice programmers. This means that this aspect of programming cannot be properly taught using dBASE. However, the concepts of tables (lists and two dimensional) used for reference can be taught by simply using files in place of internal program arrays.

Specialised data processing language.

The dBASEII language is specially designed for data processing and lacks many of the 'low level' features of BASIC. To give one example it is not possible to draw graphs using dBASE.

Languages like Pascal are considered to be 'better' than BASIC, partly because they are block structured. The dBASEII language is not block structured.

Some of these missing features are serious, and may preclude some people from using the language.

## 6.10. Summary

In this chapter I have tried to reflect my own enthusiasm for the

use of the dBASE language in a computing course. This chapter has shown that dBASEII could be used to teach the main aspects of programming concepts to computer students.

dBASE could not be used to teach programming since it has some serious drawbacks, but froms a valid tool for teaching novice programmers some of the most important programming concepts. What has been outlined is a way of introducing programming not through using a language, but through using a DBMS with a procedural language. The student will always be writing small programs to complete programming tasks which are realistic and that the student can relate to the rest of the course.

This is perhaps the most significant aspect of this approach. The student is using an environment for learning about computing concepts (the DBMS) and is using the same environment for learning about programming concepts. The course forms a coherent whole.

The student is viewing the programming task at a high level, relating the programming tasks to the system they are working with, writing significant programs rather than the usual and common 'write a program to print out the seven times table' type.

The student will naturally be viewing the system from a top down viewpoint and producing code which fits into the overall view of the system that the student will already have. In many ways the code is equivalent to the design level pseudo code of conventional languages, and Basingstoke experience indicates that students tend to develop good design skills and programming style after having used dBASE.

The language of dBASE cannot be used to teach programming in depth and the conclusion seems to be that this approach to teaching would be used to introduce novices to important concepts and approaches which would be built on later using another language. Since the student

work with the DBMS will integrate various areas of the computing course there should be time for students to progress to other languages.

This kind of approach to teaching programming seems feasible and advantageous as compared with using a conventional programming language, especially as compared to BASIC. If the DBMS could be linked to an educationally more acceptable language (perhaps PASCAL) then the result should produce a much better way of teaching programming.

## 7. Chapter Seven — Business Users Requirements

One of the objectives of the thesis was to establish the credentials for a computing course based on database systems (as opposed to BASIC) within the business community.

This research took two major directions :

To establish that databases, and especially microcomputer databases were used successfully in the business / commercial environment - particularly in an environment likely to be encountered by the students completing computer courses in the college. A subsidiary aim here was to look at the requirements of the novice business user.

To establish that the commercial users, who would be the potential employers of the students taught, required any student knowledge of database systems, and whether this knowledge was considered important, particularly in relation to BASIC.

If these commercial needs could be established, then the inclusion of a database as a major part of a computing course, would more readily be accepted by schools and colleges.

The needs were established in two ways :

1. A visit to the Automobile Association Management Services Department. This was in order to establish the way in which the AA was developing their use of database packages for end users. This includes the influences likely to made on the jobs skills required of students entering the AA as employees.

2. A survey of mostly local (to Basingstoke) firms using a

questionnaire. The questionnaire was compiled and a draft copy checked by Mike Freestone before circulating.

## 7.1. The Business Users' View

In a recent article in the Financial Times ( 7.1. Sedacca B 1985) microcomputer-based database management systems were brought to the attention of a seemingly unsuspecting business community. The Financial Times saw the need to include this article, which is aimed at the business user with no database experience, and this indicates that there was a lack of awareness of such systems. The article begins by suggesting that such systems 'can do nothing that cannot be done by a human being using a manual system' but that such a 'boring and time-consuming job is not attractive to human beings and computers can do the job much faster, albeit in a half-witted way'.

The article then outlines basic filing packages from the point of view of the manual card index systems, and then goes on to detail facilities available in dBASE II and DELTA (which is considered to be the main rival to the Ashton Tate product). Mention is made of Mr. Codd who 'postulated the relational model'. The article also states that 'It is only in recent years, thanks to the astounding success and popularity of the IBM Personal Computer, that non-computing managerial staff has taken an interest in database technology. Before then it was the preserve of data processing technocrats. Now a database management system is just one of a number of microcomputer based productivity tools, such as word processors and spreadsheets. A DBMS is useful for developing applications which are not clearly defined and regulated, as opposed to applications such as accounting..'.

This article reflects the generally perceived increasing awareness amongst the business community of the uses to which microcomputers can be put, and DBMS packages in particular. It was this awareness which the survey was meant to test.

### 7.1.1. The Perceived Relational View

An article in COMPUTING ( 7.2. Foremski T 1985) details the view from the large DP Department to microcomputer relational systems. The article begins by outling the 'classic view of the rdbms as being slow and having a high memory overhead and high CPU cycle demand has now been effectively eroded through new techniques that optimize data storage and retrieval. But now relational systems face an uphill task in persuading large DP departments that have invested heavily in the older non-relational systems, such as hierarchial and networked databases, that there is a significant advantage in moving to a relational system.'

The article details the findings of a Californian based market research firm, Software Access International, who have interviewed the managers responsible for microcomputer use in 100 companies of Fortune 500 size. They found that managers were actively discouraging the use of dbms on personal computers. This is because of a perceived need to keep control at the central corporate database level, and this would not be possible with the advent of individual databases.

Many of the personal computer databases implemented were relational, although most users were not aware of this. They also found evidence of the larger minicomputer relational systems becoming available as compatible systems on microcomputers, which will also tend to offer a lever into the more entrenched corporate dbms.

In a more recent survey Software Access questioned 3,700 personal computer users at home and in the workplace, 45% had some form of dbms. The largest category was 63% with wordprocessors, and spreadsheets next with 48%.

Rob Leftkowits, a senior analyst at the market researchers, sees no

sudden growth in mainframe relational systems, because of the difficulty in moving from system to system. However, he predicts a large increase in the use of relational systems where personal computers are put on the users' desk.

The article, by Tom Foremski, concludes by indicating that IBM are suggesting that DB2 (their relational system) will replace IMS (a formatted system), and also suggesting that the marrying of relational systems with artificial intelligence systems will provide the user with an optimized system. 'The biggest challenge lies in constructing a truly distributed rdbms that can sit on different hardware while the user can access data without knowing where the data is or on what machine.'

The article suggests that relational systems will be widely used and will challenge the entrenched use of other database systems within large data processing departments. This is an experience which has been found to exist in the Basingstoke area.

## 7.2. Aston Management Centre

Professor C D Lewis ( 7.3. Professor C. D. Lewis 1985), Professor of Operations Management at the University of Aston Management Centre, was contacted to find out his own opinions on the relevance and content of a database course which would equip novice users for a business environment. He did not indicate that he presently ran any such course, but he did include a schedule for such a course, which contained the following headings:

1. Terminology: Character, Field, Record, File

2. Command Driven v. Menu Driven

3. Level of Package:
   a. Database creation and simple search
   b. As (a) but with more sophisticated search facilities and a report generator
   c. As (a) and (b) but with macro programming

4. Search criteria

5. The difference between sorting and indexing

6. The use of multiple index keys

7. Report generators and the use of indexed files to trigger sub totals on numeric fields

8. Complete or selective reports on all or selected records

9. Macro programming.

## 7.3. The Automobile Association

Many firms in the Basingstoke Area have begun to use microcomputer database packages since 1985 (and indeed before this date). Three large firms have introduced personal computers with data base packages in 1985 :

| | |
|---|---|
| Eli Lilley | - 20 pc dBASEII systems |
| Lansing Bagnall | - 10+ pc dBASE systems |
| AA | - over 50 pc installations |

All of these firms have large DP departments and have made a corporate decision to introduce pc systems. The AA was chosen for a detailed analysis because it was accessible (helpful management) and also the biggest local employer of college students.

The AA ( 7.4,Mr. Brian Harrison & Marilyn Lucas 22/4/85) have some 1500 employees based in Basingstoke, and some 200 in Management Services, which deals with computer systems throughout the organization (but not with Operations). The company is extensively computerized, with membership systems dealing with millions of members (of the AA) on computer file. As well as the motoring services offered, the AA also deals with hotel and garage classifications, insurance and other activities. Like many organization with such widespread computer usage, there is a backlog of computer projects waiting to be dealt with by management services.

### 7.3.1. AA Corporate Policy

Up to 1985 other departments were allowed to buy in pc equipment as they saw fit, and this led to several different systems being installed. However, it was always the policy of Management Services not to offer support for these systems, and the relevant departments had to solve their own problems. This led to various different directions within the company. Some sections developed systems in BASIC, others did use dBASE II.

During 1985 Management Services developed a policy whereby all hardware and software purchases should be made through them, and so standardized future purchases. The hardware chosen was the ICL PC and the IBM where appropriate.This also led to a policy of using dBASEII as well as other packages, to create a set of software which was supported by management services. This support is in the form of consultation advice, and also training (in-house) in the recommended

packages.

#### 7.3.1.1. dBASEII Systems

Details of Management Services policy and experience were given in interview by Brian Harrison, Business Systems Controller in the Business Systems Development Department of Management Services, and by Marilyn Lucas from the same department who had implemented and designed the insurance system in dBASE II.

Management services (in 1985) installed two example systems, one to deal with a computerized 'staff suggestions box', which was a complex system and had not been finalised, the other system was for dealing with insurance claims.

The insurance claims system had been written and implemented from Management Services in the traditional way, and was not developed by the user. Part of the system is shown on the next page in the form of a screen dump of one of the user input screens.

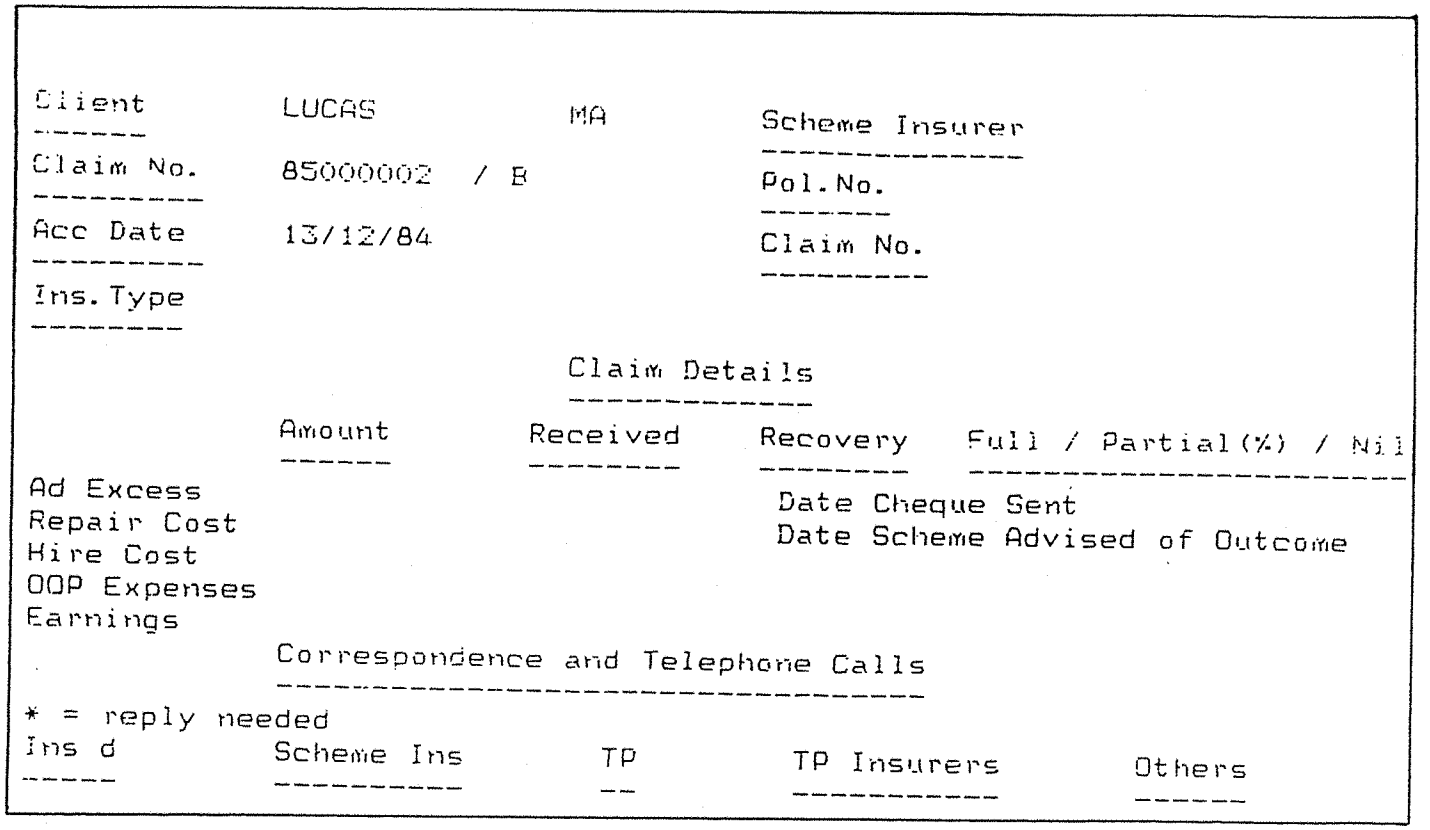



Fig. 7-1: AA Insurance System Screen

This dBASEII system was working and installed in the AA on ICL PC´s. However, the system was not performing well, because the operating system used on the ICL machines (MP/M - concurrent CP/M) did not match dBASE II requirements, and the user found that the system would crash occasionally. In itself this was not a problem, and technically the system could be simply restarted, unfortunately the crash created update problems because any ammendments or new records would be immediately stored on disc, whilst the associated index file was updated in memory (and not restored on disc until dBASE was QUIT). On restarting the system then, the file was not in sequence with the index, and had to be reindexed. Again this did not cause any user time overheads, but because of the size of the file the reindexing took some three hours, during which time the user could not use the computer. Feeling expressed was that the dBASE II system would have been very useful if it had worked in a reliable way.

#### 7.3.1.2. dBASE II Users

Clearly the AA Management Services saw the dBASE II system as not suited for end user except with a developed and implemented turnkey system. These are perceived user requirements based on experience of users and systems within the AA.

### 7.3.2. What the User Wants

Management Services have found that dBASE II is not suitable for releasing to their novice users. Many of these potential users were defined as :

Mature (I think they mean over forty !)
Having no computer usage experience at all

It was felt, probably quite reasonably, that dBASE II would prove much too complex. The systems released to these end users, for the users' own development would be SUPERCALC2 and one of various wordprocessing packages. Experience showed that users may still need the spreadsheet package to be set up for them. Before any package was released to the user two days in-house training was given.

### 7.3.3. What the User Does

The perceived requirements of the AA users does not account for the usage by some sections, where the personnel involved had prior computing experience. One example was in the insurance section where an employee had developed quite sophisticated systems using BASIC. Unfortunately these proved difficult to use when this employee left for another post.

### 7.3.4. Future User Requirements

The AA have now adopted the use of the SMART integrated package with wordprocessor, spreadsheet and filing package. Most users have found this acceptable.

Management Services say that they would look for new employees to have a good knowledge of databases (and spreadsheets and wordprocessors) but would still prefer some knowledge of BASIC programming expertise rather than expertise in dBASE II. The reason given for this last response was simply that BASIC was the normal experience found, and a departure to a system like dBASE II would be radical and thereby possibly not successful. However, Management Services did not seek to suggest that BASIC skills were actually a good thing either.

This requirement was stated in 1985. Interestingly, in 1986 the requirement stated in the questionnaire had changed away from BASIC. This change is perhaps the strongest case for a shift in the attitudes of local employers away from the traditional desire for BASIC - because its always been that way, and towards skills in systems like dBASE II.

## 7.4. User Views Survey - Introduction

A survey of some fifty firms in Basingstoke, was carried out in November 1985 to January 1986. The survey was split into two main sections, the first part dealt with the perceived requirements for training in general, together with the number of personnel employed in the company. The second section deals with the database questions which were compiled in collaboration with Mike Freeston. The questionnaire was issued from Basingstoke Technical College as part of the requirements for a local area survey of requirements for a new BTEC HND course in Commercial Data Processing. The inclusion of the

thesis survey was fortuitous and happened to coincide with some of the needs of the new College course. The HND course contains several modules which relate to database systems.

The following pages detail the survey documents, analysis procedures. Some of this information originates from another system and has been pasted into the thesis.

## 7.5. The Survey Letter

Each company surveyed received an introductory letter which outlined the aims of the questionnaire. The choice of companies was based around a list of firms in the Basingstoke area currently known to the College. The letter is printed below.

Computing Section,
Basingstoke Technical College,
Worting Road,
Basingstoke.
Hants.
RG21 1TN.

Basingstoke Technical College is introducing the new BTEC Higher National Certificate in Business Data Processing in September 1986. This course will be designed to serve the requirements of industry for Business Data Processing training, and will be in the form of a day release or evening course of two years duration.

We would like your help in adapting the design to fit your needs. An outline of the proposed course is enclosed, together with a brief questionnaire which we would like you to complete. The questionnaire is designed to enable you to influence the course design so that students studying the course would be provided with professional skills relevant to your own organisation.

We would appreciate the return of the questionnaire by Wednesday 27th November. If you have any queries then please phone me on Basingstoke 54141 extension 40.

Mr. S. Hitchman.
Computing Section.

## 7.6. User's Survey Introduction

In addition to the letter an introductory description of the proposed HND course was sent. Included in the outline are details of the modules concerned with databases in the proposed course. The introduction is printed below and on the next page.

1. Introduction

This course aims to provide a broad base for students requiring skills development in a commercial data processing environment. It is not, however, confined to traditional large-scale centralised business systems, but extends into database management systems, small business systems, and distributed systems.

Potential students should be interested in a range of areas working as commercial programmer, small systems manager, business analyst or database administrator/programmer. The course concentrates on the basic skills of programming, systems anlysis and design within the coherent framework of the business environment, its particular needs and available computer-based system aids.

Students must become familiar with a range of equipment including simple networks or distributed systems. They should also become competent in the use of business packages and database management systems.

2. Structure

| | |
|---|---|
| Year 1 | Applications for Business |
| | Information Analysis 1 |
| | Computer Systems Architecture |
| | Programming |
| Year 2 | Business Information Systems |
| | Information Analysis 2 |
| | Information Systems Architecture |
| | Software Implemenation |

3. Content

Applications for Business is the main integrating unit in year 1, and is based upon case studies which graduate from a simple business system using packaged software to tailor-made systems for larger organisations, including an introduction to databases. The student progresses through an analysis of requirements, requirements specification and simple system design, to an understanding of the main components of computer-based business systems and their

development stages. The unit will introduce systems design methods which will be formalised in year 2. All three other units in year 1 will contribute directly to this unit, giving the necessary background knowledge for case studies and simultaneously providing experience in the use of particular techniques.

Similarly, the second year unit Business Information Systems is central to the whole course and will draw from the other units as the year progresses. It extends the students' knowledge of organisational needs and develops a structured approach to the design and implementation of a computer-based information system. In particular, it will provide the framework for the development of a case-study based project integrating all the final year units.

Information Analysis progresses over both years of the course and provides a vehicle for integration since it consolidates the necessary skills in quantitative methods, business organisation and communication relevant to the development of a requirements analysis for an information processing system.

Computer Systems Architecture provides for a basic understanding of hardware and incorporates the quantitative methods necessary for a sound understanding of the operation of a computer-based system. The emphasis, however, is on the applications of hardware, and so a variety of systems will be considered with a view to their relevence for particular business information systems. Simple techniques for data communication will be taught as an introduction to the concept of distrubuted systems within the second year unit Information Systems Architecture.

Information Systems Architecture looks at more advanced systems and, in particular, distributed systems and databases. This unit aims to develop the student's understanding of the relevence of particular systems implementations to a specific application or organisational environment. The student will be expected to use a database and develop applications for a distributed computer system.

Programming is taught through the two years of the course. In year 1 the Programming unit will develop a thorough understanding of programming principles rather than a mastery of one language. This unit introduces a highly structured commercial language to explore the range of facilitiies provided by current languages and may be taught through the use of a database management system. The second year unit Software Implementation is more particularly concerned with developing expertise in commercial programming but also extends this study to encompass systems software, language comparability and operating systems.

## 7.7. Blank Questionnaire Form

The questionnaire form was kept a simple as possible, in that most answers required a simple tick, or choice of alternatives. A blank form is printed on the next four pages.

Part 1

1.1 How many people in your organisation work with computers ? _____

1.2 How many of these are involved in developing computer systems ? _____

1.3 How many of the above need training ? _____

1.4 Which of these training areas would you require ? (Please tick those requred)

- Information Systems Analysis _____
- Information Systems Design _____
- Information Systems Testing _____
- Computer Systems Hardware _____
- Selection of Equipment (Hardware) _____
- Selection of Software _____
- Business Application Packages _____
  - Wordprocessing _____
  - Spreadsheet _____
  - Database _____
  - Database Management Systems _____
  - Financial Modelling _____
  - Statistical Analysis _____
  - Any Others ? (Please Specify)

In-house Application Packages _____

Programming (Appreciation) _____

Programming (Advanced Skills) _____

Distributed Systems _____

Networks _____

1.5 Would you prefer the emphasis to be placed on :

Microcomputers _____

Minicomputers _____

Mainframe computers _____

1.6 Have you any other perceived requirements ?
(Please Specify)

1.7 Does your organisation normally :

Hire qualified DP staff _____

Train your own DP staff _____

Send your employees to train with suppliers of hardware and software, or with government trainers _____

1.8 Would your organisation support this new course at BTC as part of its training and recruitment policy ? _____

1.9 Any other comments ?

Part 2

Database Skills and Experience Requirements

The need for students to have aquired skills and experience in handling and developing databases (such as practical experience in developing systems using Microcomputer Database Management Systems - MicroDBMS) seems to be increasing, and could be emphasised in this course. These questions are designed to see if this need can be established and should be covered by the course

Please circle the appropriate response:

2.1 Does your organisation use computer systems with database packages.

YES / NO

If YES, then which one(s) ?

.....

2.2 Do you employ a Data Base Administrator / Manager ?

YES / NO

2.3 Do you think that a good working knowledge of typical Microcomputer Database Management System (MicroDBMS) is:

NOT NECESSARY / USEFUL / ESSENTIAL

If USEFUL or ESSENTIAL, which package would you recommend ?

.....

Do you need help in evaluating such packages ?

YES / NO

2.4 Do you think it is important for students on a specialist computing course to have a good working knowledge of a MicroDBMS ?

YES / NO

2.5 Most College students completing courses in Business Studies take computing as part of their course. Do you think it is important for these students to have a good working knowledge of a MicroDBMS ?

YES / NO

2.6 If you had to choose between two prospective employees, one with a good working knowledge of BASIC and the other with a good working knowledge of MicroDBMS, who were otherwise equivalent, which would you chose ?

MicroDBMS / BASIC / Don't Know / Not Applicable

2.7 Do you think it is preferable to learn to program in the command language of a MicroDBMS (for example dBASE II) rather than in BASIC ?

YES / NO

PLEASE STATE REASONS for your response to question 7:

ANY OTHER COMMENTS REGARDING DATABASES:

(End of Part Two - Part Three is over the page)

Part 3

Concluding Information

3.1 Please add your name, department, and phone extention :

3.2 Would you like to participate further in implementing this course, or in receiving it ?
(Please Specify)

End of Questionnaire

Thank you for spending time in completing this questionnaire. It will help the College in our mission to design better courses for our clients.

## 7.8. Analysis Procedures

The details of the analysis procedures, which involved the use of dBASEII, are fully discussed in the appendix.

The survey responses were keyed into dBASEII files and command procedures with reference files were used to analyse the results. Most of the questions elicited between two and four alternative responses to the choices offered. The totals for each question where then entered into a graphics package which was used to produce bar charts to summarize the question responses.

The next section does refer to the command and data files which were used in the dBASEII system. The ANALYSIS command file was the main procedure, and produced and collated the results from all of the survey files.

The analysis was completed in two parts, part one consisting of 25 simple yes/no questions which are summarized with a stacked bar graph. Part two of the questionnaire consisted of various questions relating to the use of databases, and a bar graph is produced separately for each question. In addition, some questions resulted in a list of responses, and for these questions the resulting lists are shown.

The use of dBASEII for the analysis greatly facilitated the production of the survey results, and once the system had been set up it was quite simple to add in new responses and produce interim results. This again reflects the versatility of database packages and the many uses that they can be put to.

## 7.9. Details of Survey Results

The printed output from the ANALYSIS command file was used to put data into a spreadsheet package, and so produce bar charts. The results of the analysis are detailed in the following sections.

### 7.9.1. List of Employees by Company

This list is useful because it establishes the size of the companies involved in the survey.

ANALYSIS FOR PART 1 OF THE QUESTIONNAIRE

PAGE NO. 00001
17/02/85

| COMPANY SURVEYED | TOTAL EMPLOYED | TOTAL IN DP | TRAINING NEEDS |
|---|---|---|---|
| METRON | 13 | 1 | 7 |
| MUCON | 15 | 5 | 3 |
| Aspin Management Systems | 7 | 6 | 6 |
| Brown & Root Ltd. | 500 | 50 | 50 |
| Chas A Blatchford & Sons | 30 | 2 | 30 |
| Douglas Engineering | 2 | 0 | 0 |
| Eli Lilly & Co Limited | 300 | 40 | 40 |
| Hartley Measurements | 12 | 2 | 0 |
| I T W Limited | 30 | 3 | 3 |
| Macmillan Limited | 100 | 3 | 0 |
| Snamprogetti | 100 | 8 | 0 |
| Sony Broadcast Ltd. | 100 | 15 | 15 |
| Spring Grove Services | 300 | 8 | 2 |
| TSB Trust Company | 150 | 70 | 70 |
| Technicon | 100 | 3 | 2 |
| Thomas de la Rue & Co Ltd | 30 | 2 | 0 |
| Thompson - CSF | 25 | 1 | 1 |
| Transamerica Instruments | 28 | 3 | 6 |
| Wicks & Wilson Limited | 15 | 1 | 0 |
| Automobile Association | 1500 | 150 | 30 |
| Lansing Bagnall Limited | 150 | 80 | 80 |

## 7.10. Part 1 of Questionnaire

This part of the questionnaire is of only passing interest from the thesis point of view, and is included for completeness. Some of the questions are of interest. Questions 10 and 11 refer to training needs in database and database management systems.

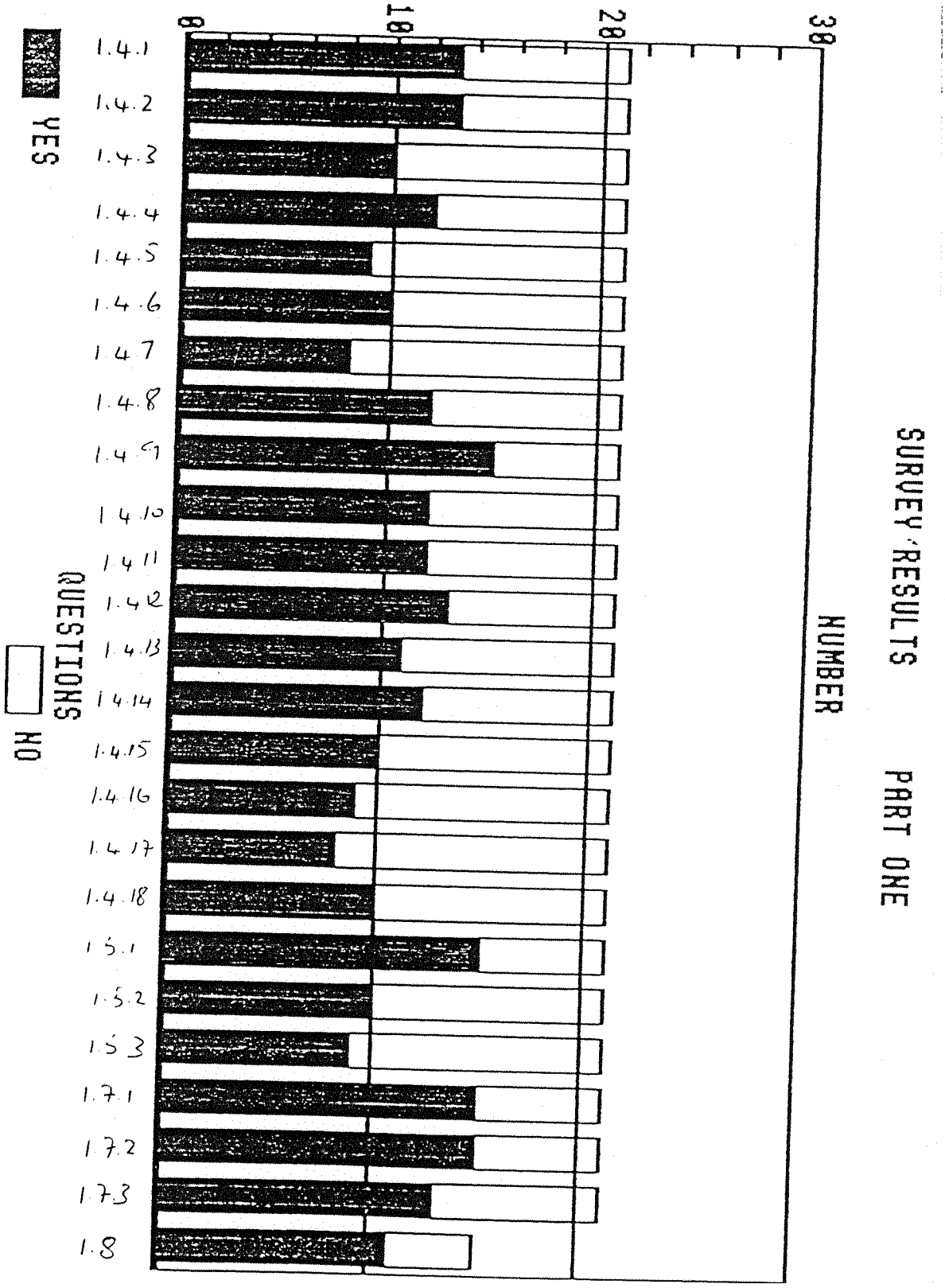
### 7.10.1. Bar Chart of Responses

The bar chart shows the total number of Yes / No responses to each question. The responses to question 10 and 11 show that roughly half of those responding would be interested in receiving training in databases and database management systems.

Interestingly question 9, which relates to spreadsheets, shows the highest response. This indicates that spreadsheets are perceived as being very useful. Suprisingly distributes systems in particular, and also networks, are not seen are training needs, despite their popularity in 1985/86. Microcomputer skills are seen as important by most people, and minicomputer and mainframe systems are not so popular. This may reflect the image of the College within the community, as not being able to provide detailed knowledge of particular systems, whereas microcomputer software skills are seen as more portable.

Question 1.4.7 (Business Application Packages) caused some confusion, it was not clear whether this was a sub heading for the packages listed on the questionnaire. The results for this question should be ignored.

Very few respondents mentioned any other requirements (question 1.6).


SURVEY RESULTS PART ONE
NUMBER
0
10
20
30
YES
NO
QUESTIONS
1.4.1
1.4.2
1.4.3
1.4.4
1.4.5
1.4.6
1.4.7
1.4.8
1.4.9
1.4.10
1.4.11
1.4.12
1.4.13
1.4.14
1.4.15
1.4.16
1.4.17
1.4.18
1.5.1
1.5.2
1.5.3
1.7.1
1.7.2
1.7.3
1.8

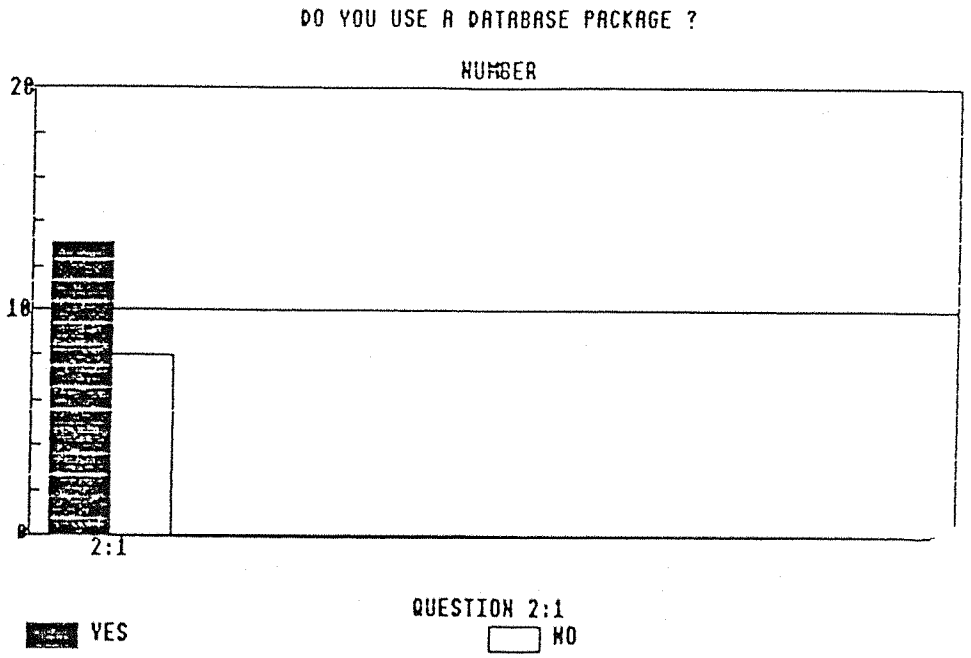
## 7.11. Part 2 of Questionnaire

Each question is presented separately, together with comments. A bar chart is used to show the analysis of responses.

### 7.11.1. Question 2.1.1

Does your organization use computer systems with database packages ?

This question was included to ensure that the respondents had a reasonable level of knowledge of database packages. Most respondent used database packages.



### 7.11.2. List of Database Packages Used

This list is taken from responses to question 2.1.2. Companies can be identified by their record number if required.

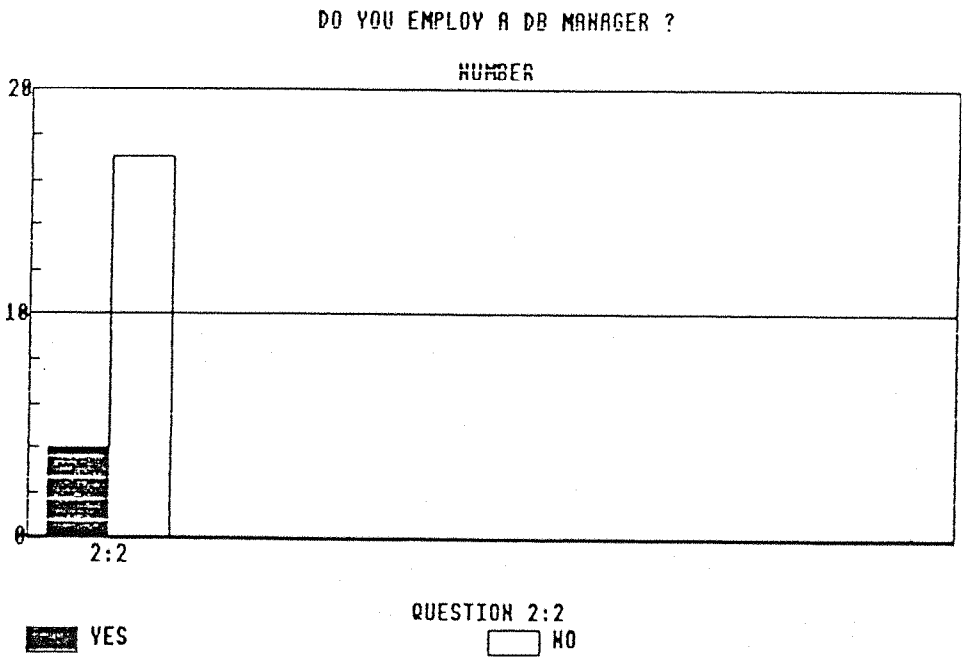
```
LISTING OF USERS DATABASE PACKAGES (QUESTION 2.1.2)

00001  DELTA 4 ON WANG PC
00002
00003
00004  IMS,NOMAD,ORACLE
00005
00006  PAYROLL
00007  TOTAL,10/22,ETC.
00008  COMPSOFT DMS/DELTA
00009
00010  DBASE II
00011  DBASE II
00012  DLI
00013  IDMS
00014  DBASE II , POLISY , PALM
00015
00016  DBASE III
00017
00018  SYMPHONY , DELTA
00019
00020  ICL DRS20, DGMV8000, ICL VMES, dBASEII, dBASEIII
00021  IDMSX, dBASEII
```

### 7.11.3. Question 2.2

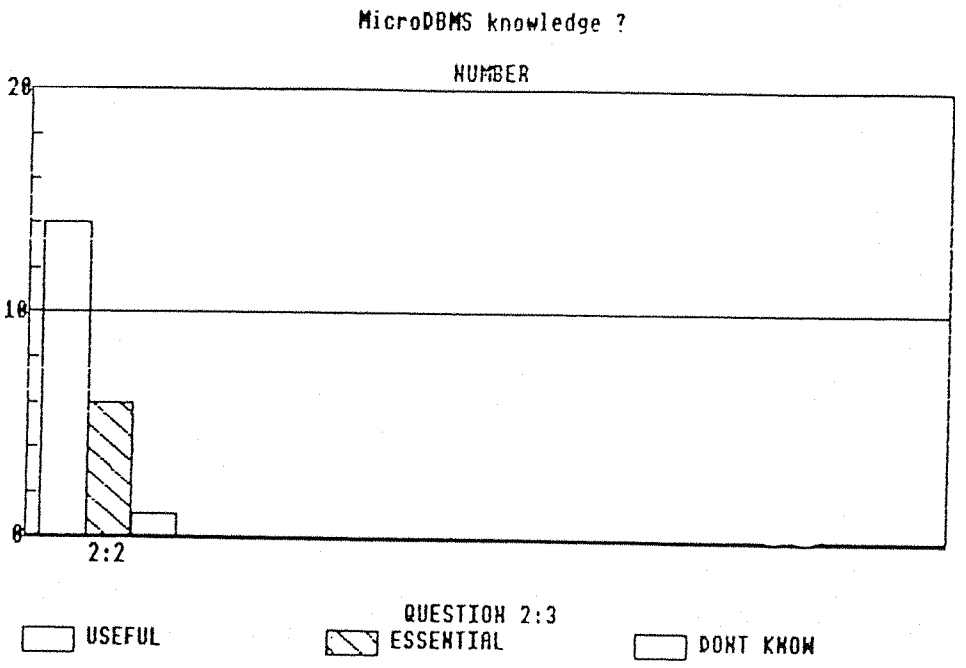
Do you employ a database administrator ?

This gave a very interesting response, an overwhelming no. Probably indicates that the companies concerned are still tied into traditional DP systems.



### 7.11.4. Question 2.3

No respondent thought the knowledge to be not necessary. Most considered it to be useful, although there were a good number of essential responses. (There was 1 don't know)



### 7.11.5. List of Users Recommended Packages

This list is taken from question 2.3.2 and users have been asked to comment on which package they would recommend for microcomputer use. This question was introduced for interest only, but could also establish a requirement for dBASE II.

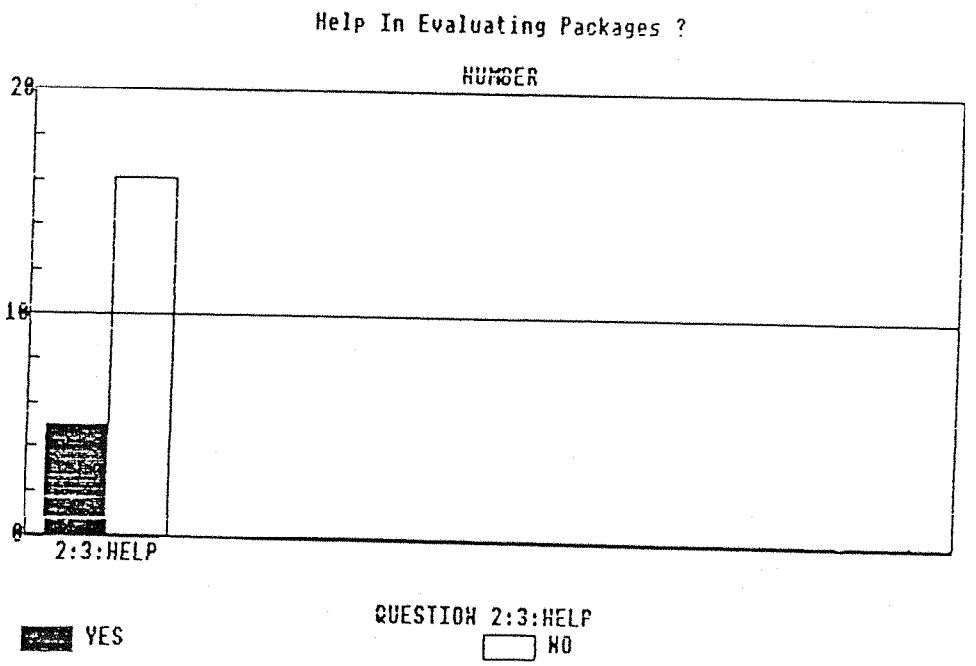
LISTING OF USERS RECOMMENDED DATABASE PACKAGES (QUESTION 2.3.2)

00001 NO EXPERIENCE
00002 DBASE II
00003 NOMAD OR dBASE II
00004 DON'T KNOW
00005
00006 DON'T KNOW
00007 DBASE II
00008 DMS/DELTA
00009
00010 DBASE II
00011 DBASE II
00012 OPEN ACCESS
00013 DBASE II
00014 DBASE II , SMART
00015 DON'T KNOW
00016 DBASE III
00017 NO OPINION
00018 DELTA OR DBASE
00019 DON'T KNOW
00020 WHICHEVER IS BEST !
00021 DEPENDS, RELATIONAL FOR MICRO, OTHER FOR MAINFRAME

### 7.11.6. Question 2.3.Help

Do you need help in evaluating such packages ?

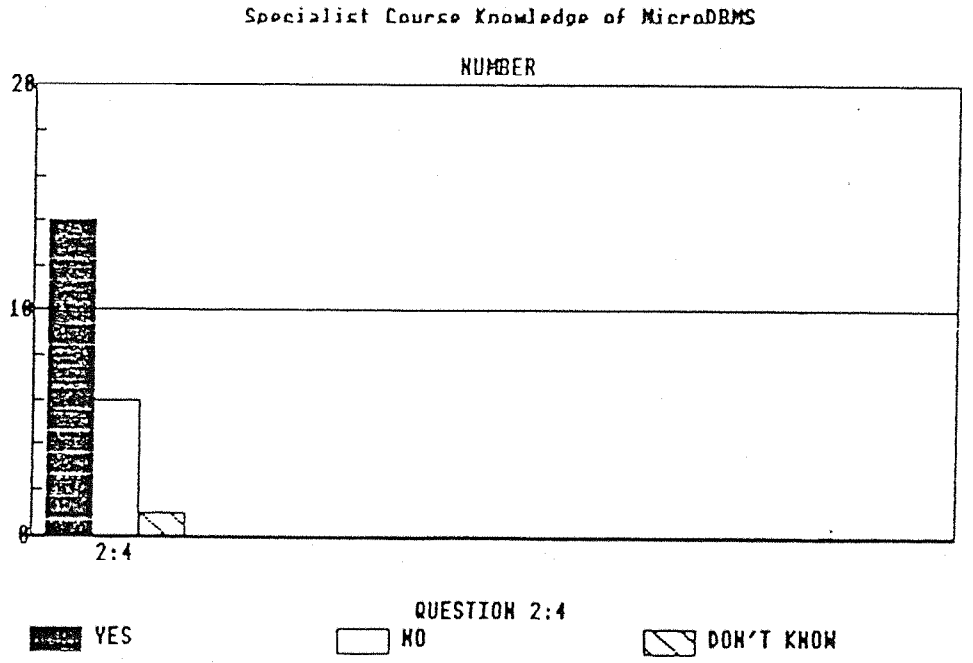
This question was designed to see if the respondents felt that they knew enough about evaluating such packages. The overwhelming no response indicates that the respondent´s other answers were based on experience.



### 7.11.7. Question 2.4

Do you think it is important for students on a specialist computing course to have a good working knowledge of a MicroDBMS?

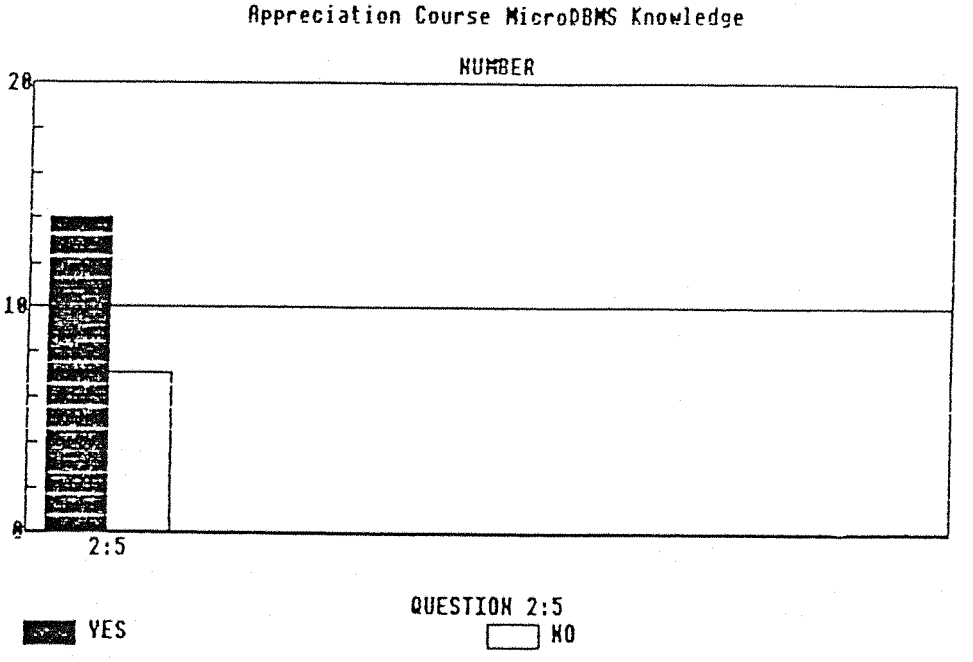
This was one of the key survey questions. A large majority in favour, but a sizeable proportion against. Tends to favour the thesis argument.



### 7.11.8. Question 2.5

Most College students completing courses in Business Studies take computing as part of their course. Do you think it is important for these students to have a good working knowledge of a MicroDBMS ?

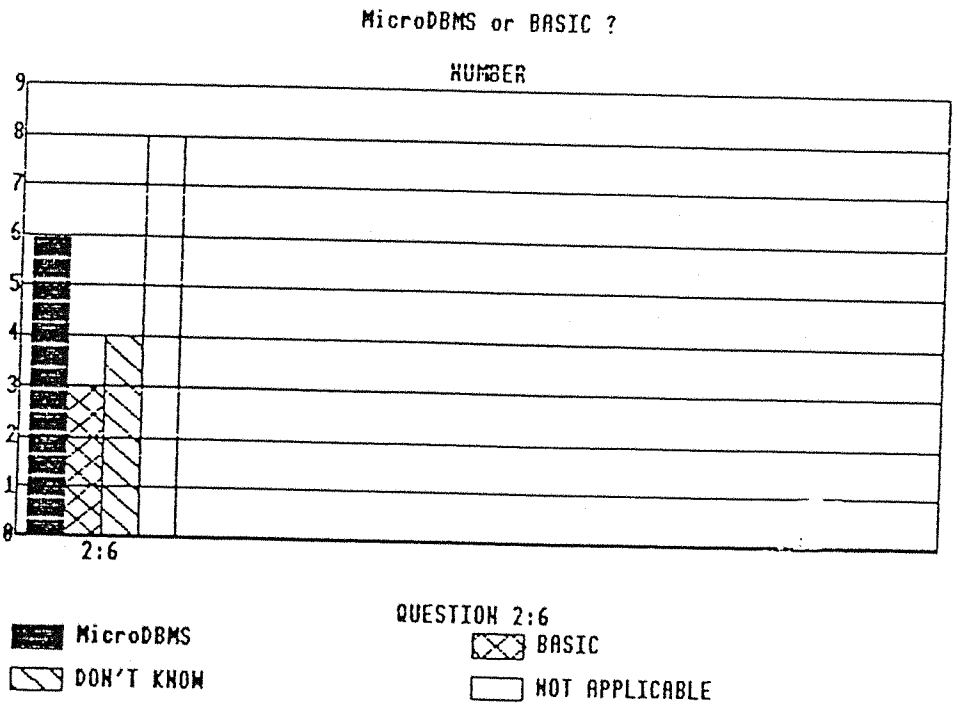
Again a key question, designed to establish a need on lower level courses where the computer content would be appreciation rather than specialist. This ellicited the same response,with a large yes majority. Again this tends to confirm the thesis argument.



### 7.11.9. Question 2.6

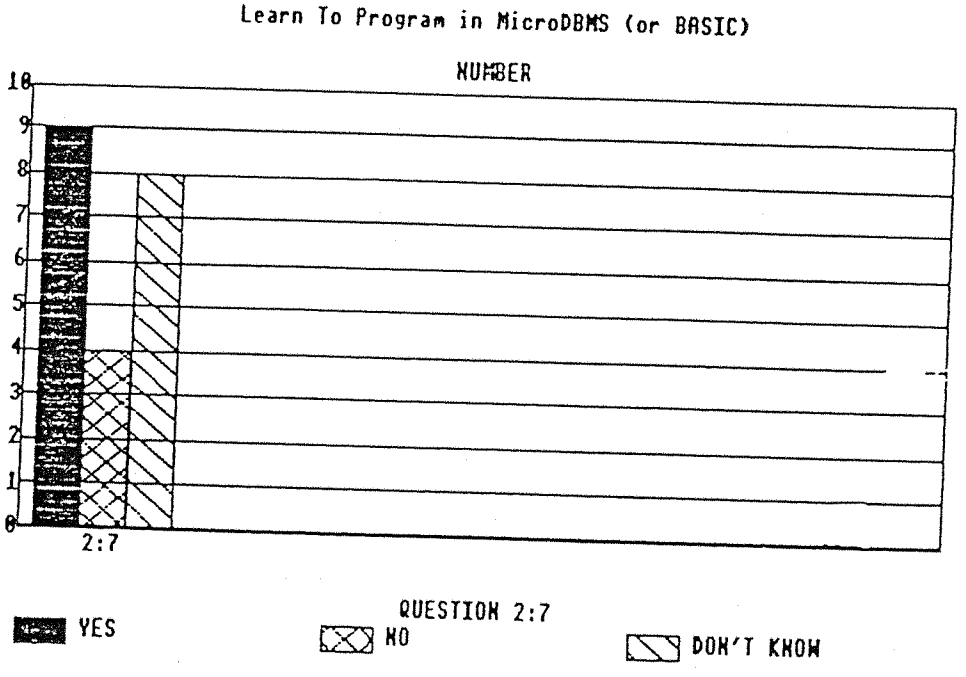
If you had to choose between two prospective employees, one with a good working knowledge of BASIC and the other with a good working knowledge of MicroDBMS, who were otherwise equivalent, which would you choose ?

This question was designed to establish the BASIC / DBMS preference (especially with regard to programming skills). The biggest response here was not applicable, which may indicate that neither is a valid qualification. This probably reflects the need for skills in languages such as COBOL. However, there were six respondents in favour of a DBMS and only 3 for BASIC.



### 7.11.10. Question 2.7

This question reinforces the previous one, and respondants were mixed almost evenly (with four don't knows). The fact that so many choose the DBMS tends to support the view that there is a commercial basis for teaching structured programming this was, perhaps as preparation for 4th generation systems.



### 7.11.11. Dbase or BASIC - Reasons

This is a list of responses to the question which asks for some explanation of the answer to 2.7. Some of the responses have been shortened.

LISTING OF USERS REASONS FOR QUESTION 7 (DO YOU THINK USE OF DBASE II ..)

```
00001  D
00002  M DBMS KNOWLEDGE OF MORE VALUE TO THE COMPANY
00003  N DBASE II WOULD BE TOO SPECIFIC
00004  Y BASIC RESULTS POOR, POWERFUL ROUTINES IN DBASE
00005  Y
00006  D NOT ENOUGH KNOWLEDGE
00007  N TOO EASY TO LEARN
00008  D
00009  Y MICRO DBMS MORE LIKELY TO BE USED IN COMM ENVIRON
00010  D DEPENDS ON THE JOB SPEC
00011  Y COMMERCIAL APPLIC SUITABILITY (BASIC ISN'T)
00012  Y USERS MUST LEARN THAT LANGUAGES ARE TOOLS TO SOLVE
00013  N REASONS DON'T SEEM SENSIBLE TO ME
00014  Y BASIC HAS NO COMMERCIAL APPLICATION IN COMPANY
00015  D STRUCTURED PROGRAMMING GROUNDING NECCESSARY
00016  Y PC USERS USE PACKAGES, DON'T NEED PROGRAMMING
00017  D
00018  Y DBMS ARE USER FRIENDLY AND USERS DON'T NEED PROG
00019  D IF YOU CAN USE BASIC CAN PICK UP ANYTHING
00020  N HIGHER LEVEL THAN BASIC & GENERATES QUICKER RESULT
00021  Y WANT SKILLS IN EXPLOITING PACK SOFT (NOT IN HOUSE)
```

## 7.12. Summary of Survey Findings

There was no overwhelming support for the use of a MicroDBMS in training, but there was a consistent response of around 50% to support it. This was better than expected, since it was considered that many professional DP respondent from large departments, may not have experience of such systems. This does not seem to have been the case, however some respondents did stress the need for languages like COBOL.

I consider that the results support the inclusion of a large element of a MicroDBMS system in computing course at Basingstoke, and I see no reason why this should not be true of other areas as well. The survey successfully established that the commercial requirement for such knowledge exists, and that the requirement of BASIC is not appreciable generally.

The traditional argument that BASIC is known and required by the business community is definitely not supported by this survey. To the extent that such a survey will always suffer from two major drawbacks:

Lack of knowledge/understanding of the system considered
Entrenched preference for the system used in-house

I consider that the survey was a success in establishing a real commercial need for skills and knowledge to be developed in Microcomputer DBMS at the expense of the traditional BASIC skills.

## 8. Chapter Eight - The Case for DBMS, a Summary.

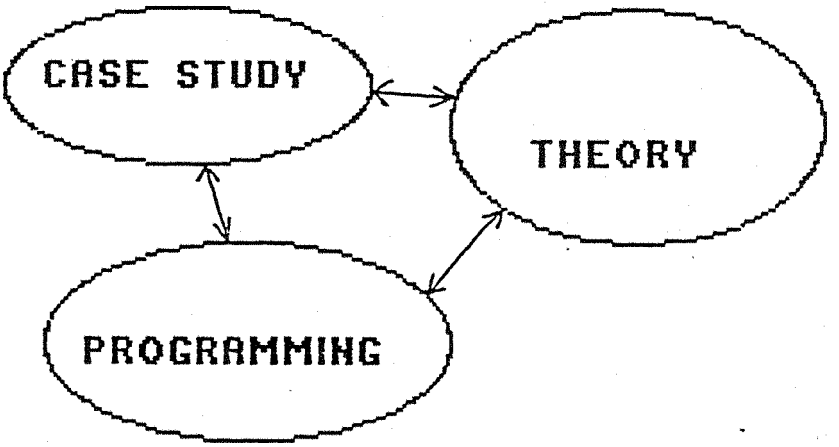
### 8.1. The Structured Approach

Most computing courses are bottom-up, in that the student is taught concepts about the subject in compartmentalised course blocks. The danger is that many students will emerge from a computing course with the impression that they have gained knowledge or skills about a set of (perhaps interconnecting but) differing areas within 'computing'. There is no overall framework within which the student can place any particular block of knowledge.

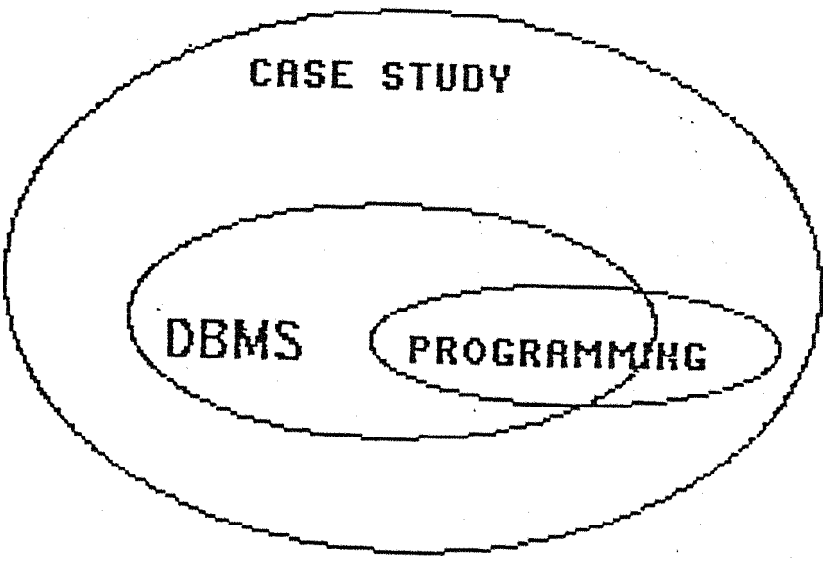
The situation has not greatly changed since the thesis was begun. Here is an example program acquired in 1987, and used by a lecturer teaching a high level BTEC Continuing Education course unit :

```
5 rem
10 dim a(50)
15 input n
20 for i = 1 to n
30 input a(i)
40 next i
50 i = 1
60 if a(i) > a(i+1) then 100
70 i = i + 1
80 if i = n then 140
90 goto 60
100 temp = a(i)
110 a(i) = a(i+1)
120 a(i+1) = temp
130 goto 50
140 for i = 1 to n
150 print a(i)
155 next i
160 n = int(n/2)
170 rem now display the median value(s)
180 print "median value(s)";
190 if n = n/2 then print a(n); "";
200 print a(n+1)
210 end
```

Obviously a 'bad' program, and ironically written in one of the most structured and powerful versions of BASIC currently available. After many hours of teaching using BASIC it is difficult to assess what, if anything, the student has acheived. This is an example of programming as an end in itself and not related to the rest (smaller part) of the student's course. The strands of the student's course are seperated:

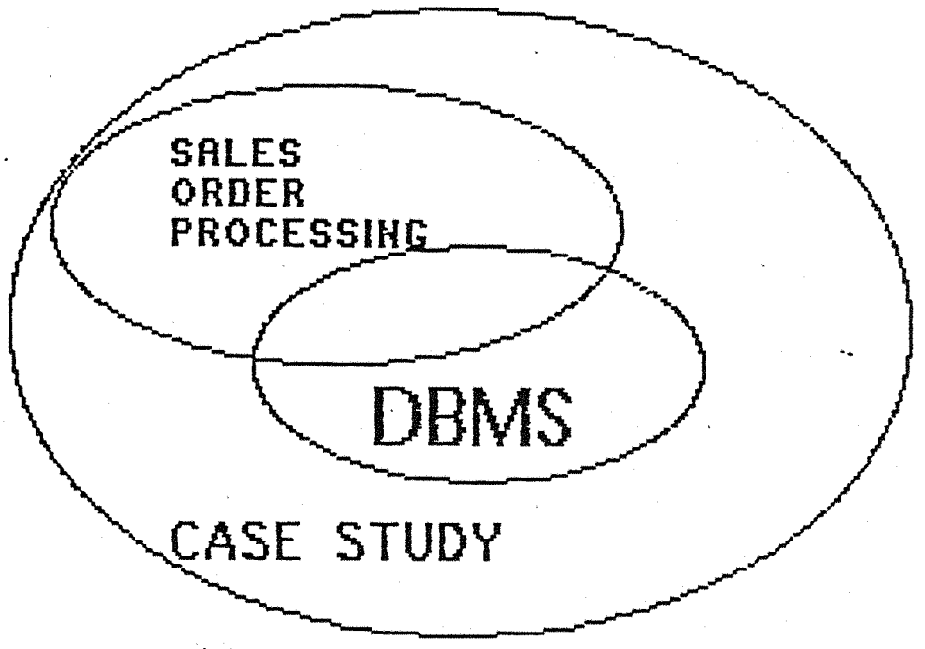



This thesis has outlined a different and new approach which would structure the course to eliminate this kind of pointless programming exercise. The thesis has shown that a course using a DBMS could span most, but not all, of the objectives of a typical computing course and that course would tend to be holistic in nature. The conclusion is that the core of the course would include the DBMS :



The proposed course should not be based on the DBMS, since this will encourage the DBMS to be seen as an end in itself. The proposed view of the computing course would be to use the DBMS as one of the tools in which to teach the student. The DBMS is viewed as an important tool, and not as an end in itself.

The diagram below shows the way in which any course aspect would overlay core parts of the course.




The thesis has discussed a set of tools and shown that these could satisfactorily form the core of a successful computing course, which would span most of the objectives of a typical course.

### 8.1.1. Acceptibility Of A DBMS Approach

At the start of the thesis research there was much interest in relational DBMS, and this interest has increased. IBM have encountered problems with DB2 (8.1. McCrone J 1986) and there is still a reluctance to use relational systems for transaction processing. Messrs Codd and Date (8.2. Not Attrib 1986) are currently giving relational systems seminars in Europe which aim to establish that

relational technology is here to stay as the basis for full production systems.

Oracle, the relational database management system company, have reported a rise in revenue of 170% for the 1985 calendar year. Oracle now market a system that runs identically on minicomputers and micros.

In Datamation recently, an article on the future of databases (8.3. Cutice R M & Casey W 1985) points to the 'commercial emergence of database languages based on abstract data types, particularly as a result of collaboration between workers in the field of databases and AI knowledge representation.' However the article pointed out that the use of relational systems was still restricted in larger transaction oriented systems.

The training needs for information technology are still unclear. Jenny Mill (8.4. Mill J 1986) questions the relevance for paper qualifications in relation to the recent need for stock market and allied systems. The experience of the potential employee in very specific areas was the major and often only consideration in job recruitment.

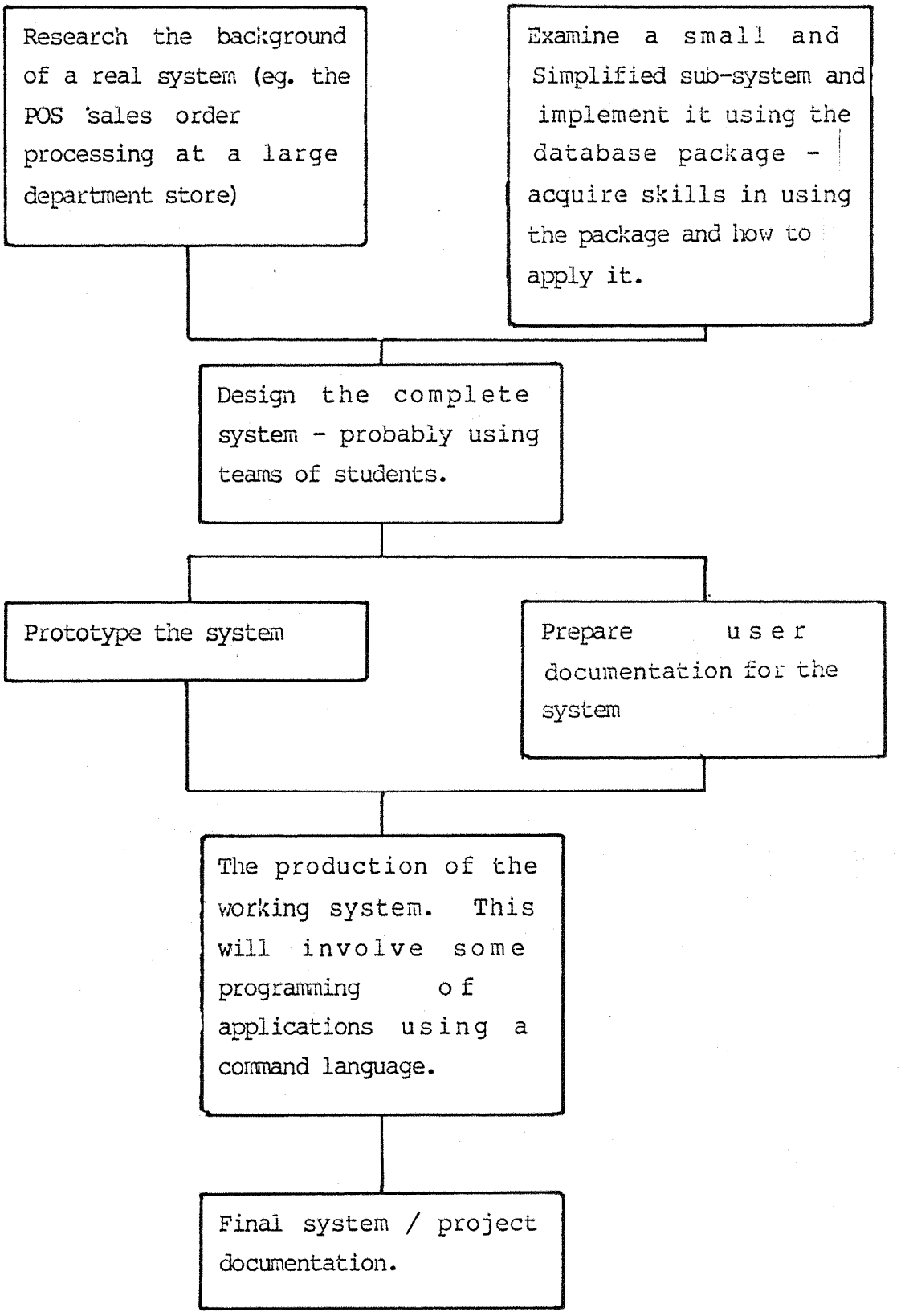
The survey detailed in chapter seven leads me to conclude that students exposed to dBASE II (for example) will be well regarded by potential employers. This has also been the experience with the students leaving the Basingstoke course who found employers attitudes positive in this respect. What should the teaching strategy be ?

## 8.2. Teaching Strategy

Recently Comal, which had seemingly disappeared, has been adopted by the Scotish Examination Board as its choice of programming language for their new O-Grade computing courses (8.5. Wardropper J 1986). The choice of language, and the general syllabus has been criticized

by those who would prefer Logo. Steven Sharpe, an examination officer for the board defended the choice by saying 'Comal was chosen because it allows a greater number of things to be done quickly'. It is unlikely that the board considered a DBMS system, but that definition of needs would apply much more to a DBMS. It is often the case that the main consideration in the computing syllabus is the programming and the language to be used.

A strategy determined by a DBMS should imply that the student is taught, and experiences a typical DBMS project life-cycle :


Research the background of a real system (eg. the POS 'sales order processing at a large department store)
Examine a small and Simplified sub-system and implement it using the database package - acquire skills in using the package and how to apply it.
Design the complete system - probably using teams of students.
Prototype the system
Prepare user documentation for the system
The production of the working system. This will involve some programming of applications using a command language.
Final system / project documentation.

The central concept to emerge is that the use of tools, including the DBMS determines and defines the course of the study towards the application of computers to solve commercial problems, and so defines the teaching strategy. Although the DBMS is not the end in itself, it does define the teaching strategy. How would this work with an example computing syllabus ? The findings relate to all of the computing courses outlined in the thesis aims (page 4). The following section details an example.

## 8.3. GCSE Syllabus

It seems appropriate to consider the thesis ideas in relation to the latest educational offering for schools, the GCSE syllabus for Computer Studies.

There are several of these, although the different English boards have worked to a common strategy. The one chosen is that from the Southern Examinations Board, since it became available to the author during research for the thesis (8.6 Southern Examining Group 1986).

The course is different from previous 'O' level courses because the syllabus is designed for a wider ability of students. The 'O' level was aimed at the top 20%, the GCSE at perhaps the top 70% of the ability range. The syllabus has been designed to be implemented around a typical computer system, such as sales order processing, and which will vary each year. This immediately gives a unifying aspect to the course, and encourages the holistic approach. Further there is a reduction in the importance of programming, and for the first time students can use a software package as the basis for their project work, rather than using a computer language such as BASIC.

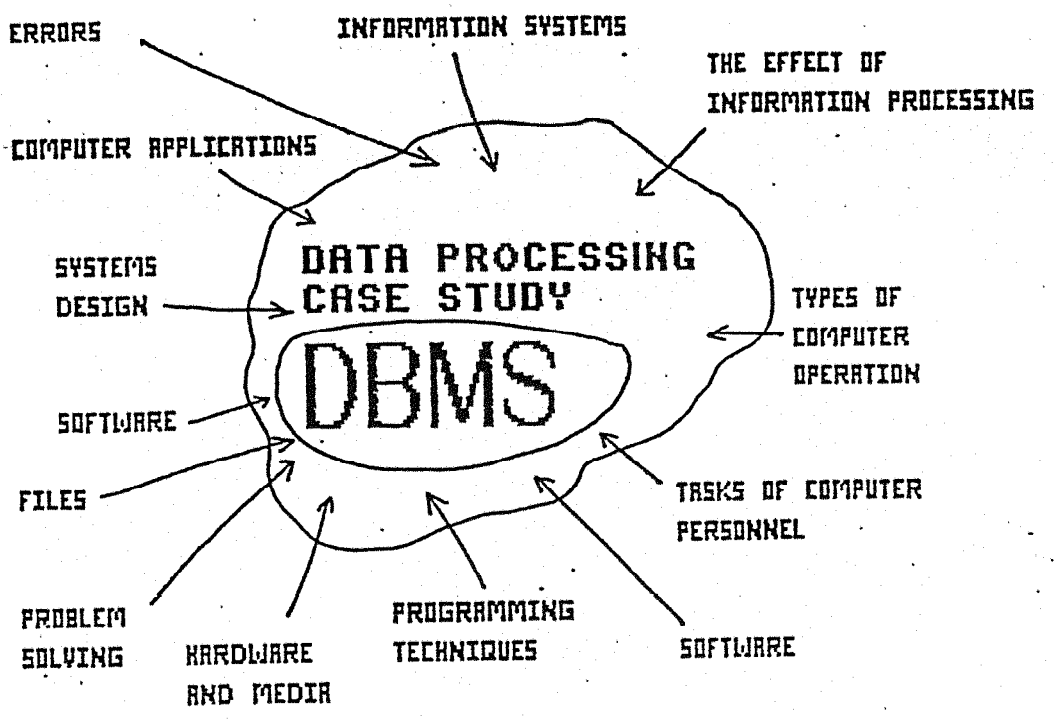
There is concern for the lack of funding, training and preparation in the trail of the teacher's dispute (8.7. Gordon A 1986). However the shortage of properly computer trained staff seems to be the major

concern.

From the point of view of the computing syllabus, the move away from programming and towards student experience and knowledge of a particular system, has opened the door to the use of a database package as the core of the course, without recourse to BASIC, Pascal, or any other language. Indeed, the teacher who avoids programming is likely to do better since more course time can be spent on relevant course material. Rather than being an alternative to usual teaching tools, the DBMS has become a major contender for this syllabus.

The implementation should be simply based around the case study, and using the typical commercial 4th generation development approach (for relational systems). The important part of the strategy is the production stage - a working system for the case study under investigation. It is at this point that the students will have come to learn about the case study and the computer systems through implementing a working system to their (user) requirements.

The emphasis throughout the course would be on design, and relating the skills and knowledge gained to case studies that the students can relate to, and will have seen and researched for themselves. Most of the syllabus content can be taught through the approach discussed in the thesis, as illustrated in the diagram below:




The three syllabus areas which do not readily fall into the approach are listed below the diagram, and would need to be taught using other approaches.

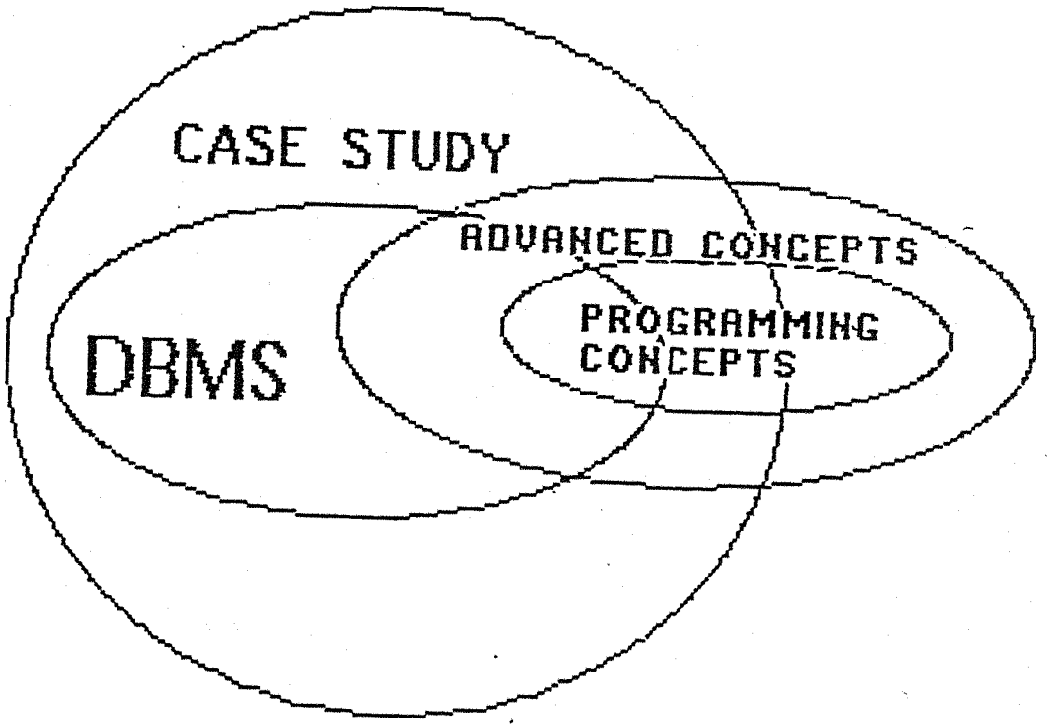
There is one major problem. The Syllabus Aim 8 (page 2) states that '...the development of skills.....including a working knowledge of a high level general purpose computer language.' No DBMS can be considered to fullfill this aim (although REVELATION, using PICK BASIC comes closest). The conclusion is that students who use, for example dBASE II, would need to progress to some other language. The conclusion from the Basingstoke experience, and as set out in the thesis, is that the dBASE experience will aid program skills with the next language.

### 8.3.1. Is dBASEII acceptable ?

The package dBASEII has been used at Basingstoke with some success. The thesis discussion has shown that careful use of the package can meet many syllabus aims, but the major drawbacks of the language leave

a large question mark over its effective use, particularly in the hands of teachers who are not fully experienced in the use of such systems.

It has been shown that the use of the package should be limited to some well defined goals. The programming language is not suitable to teach many aspects of programming, and 'further programming' can be seen as follows:




There are pitfalls, and programming tasks need to be restricted to those which require simple programs, although the power of the language being used does not mean that the programs will be simplistic in effect. The carefully designed use of dBASEII could provide an ideal introduction to some programming concepts.

As with BASIC, a tool like dBASEII can be abused in the wrong hands, and would require a skilled teacher to design a structured course.

The best result obtained with using dBASEII as an introduction to programming was that the students learnt to concentrate on design and associated techniques, and that this skill was carried over by the students in other programming languages.

It is not suprising to find that the dBASE II language is not suitable for educational use. Ashton tate did not intend to produce an educational system and they have not done so. What is suprising is that so much can be achieved using the package. It is the case that the thesis has demonstrated the potential for the approach, having found that dBASEII itself is not quite up to the task.

Having now had time to use the newer dBASEIII, which has many advantages over the older package, similar drawbacks can be found at many levels, and for the same reasons, dBASEIII is not designed to be used in schools and colleges to teach computing. It is likely to be a fruitless search to find the 'perfect' system from designers who are not trying to solve the educational problem.

**8.4. Recommendations For Further Research.**

I wish to make two recommendations for further research. Firstly to suggest that it might be possible, at some large colleges, to establish an experiment by using several 'streams' of students, all following the same course, in parallel groups, who can be taught using different approaches. This was not the case at Basingstoke, but larger colleges could provide the material to carry out further research on the effectiveness of different approaches to computing teaching.

Secondly, the main conclusion of the thesis is that the approach is very promising, but that the package, and also available packages, are not designed for educational use and so prove unsuitable in some respects. This leads to the question ..... Why use programming

languages to teach programming concepts ? BASIC, which was designed as a beginners system, as implemented, has not worked. The thesis suggests that a better approach would be to use a CAL package specially designed for computing courses. Rather as cobblers have barefoot children, computing teachers do not make much use of CAL packages (look in any educational software catalog to confirm this). Yet computing teachers are in the best position to make most use of such packages. A CAL package could be designed, based on a DBMS, which would be used in conjunction with case studies to teach computing effectively. I can use the thesis discussion to say that the CAL package should:

Allow systems to be designed (support given in design stages, for example CDD) and built quickly using sequential and indexed files. The minimum features would be:

1. Allowing the system to be defined in terms of relationships, data dictionaries and process. (A good example of the style required is found in the Microsoft Sourcewriter package to generate COBOL systems).
2. Create files
3. Paint forms for data input/update allowing several files to be accessed on one form (a good example here would be the SQLFORMS package in ORACLE).
4. Format reports (the dBASEIII system is a good example).
5. Document the system.
6. Allow graphs to be drawn.
7. Application generator, rather like the dBASEIII system.

The CAL package would interface to a programming language to handle other processing, and a subset of the language could be available in the CAL package. The COBOL 85 language would be ideal, but does not fullfill the general purpose classification. Pascal seems to be the best alternative.

## 8.5. Conclusion

I have tried to convey my own enthusiasm for the DBMS approach to computing teaching and to show that this approach has much to offer. The thesis has discussed the possibilities of using a DBMS as the core part of a computing course, and so shown that this is both possible and has produced good student performance. The package dBASEII was used as an example package and is now cheaply avaiable to schools and colleges. The package proved useful if carefully treated, but with major disadvantages. The approach has much promise, and an 'educational' version of a package like dBASEII could be very beneficial to computing teaching.

**Index To Thesis References**

The cross reference numbers refer to chapters and the references are arranged in alphabetical order by author.


6.2. Allen B,Take It From The Top,Practical Computing,May 1985 p.149
3.13. Ashton Tate,dBASEII EDUCATIONAL teaching package,Ashton Tate,1984
1.1. Atherton R,BASIC damages the brain,Computer Education, February 1982 pages 14-17
6.1. ibid
2.8. Atherton R,Why Structured Programming ?,Micro User June,1983 pps. 25-26
5.1. Bleach P,The Use of Computers in Primary Schools, Reading and Language Information Centre - Reading University School of Education,1986
3.14. Bowerman R,Relational Database Systems For Micros, DATAMATION,July 1983 128-138
6.3. Bowles K,Problem Solving In Pascal,,1977
2.5. BTEC,Syllabus for Computing and Information Systems,BTEC,1981
3.9. Childs D L,,Proc. IFIP Congress (North Holland),1968 p.162
2.10. Christensen B R,Programming Languages For Beginners and The Global Challenge,Computer Education,Feb 1982 p.18
3.8. Codd E F,,Communs Ass. comput. 13.,1970 pp377
3.12. ibid
3.1. Crowe T & Davison D E,Management Information From Data Bases,,1980
2.14. Culley L,How Girls Loose Out In DP Skills, Computing The Newspaper,June 20th 1985
8.3. Cutice R M & Casey W,Databases - What's in store, Datamation,December 1985 pps 83-88.
3.6. Date C J,An Introduction To Database Systems,Addison Wesley,1981
3.7. Deen S,Fundamentals of Data Base Systems,1980
2.16. European Communities Commission. Social Europe - Supplement on New Information Technologies and the School System. Quoted and reported as 'Experts Appraise Schools' IT', COMPUTING,November 7,1985. pps 33 - 36.
2.4. F.E.U,Computer Literacy Project Documentation,F.E.U.,1984
7.2. Foremski T,The Relational Approach Encounters Resistance, COMPUTING,May 2nd. 1985 pps. 6-8
2.18. Freeman D and Tagg W,Databases in the classroom, Journal of Computer Assisted Learning,1985 1 pps 2-11.
2.17. Freeman D & Levett J,Quest in the Learning Environment: Computer Assisted Information Handling as a Tool for Learning and Curriculum Development, Education and Computing,1985 pps 163-171.
3.4. Gane C & Sarson T,Structured Systems Analysis: Tools and Techniques,Prentice Hall,1978
8.7. Gordon A,Lack of funds threatens exams,COMPUTING, May 1st 1986 p.6.
3.3. Hancox M,Data Flow Diagrams,COMPUTING THE MAGAZINE, June 13 1985 p.22, and June 20 1985 p22.
7.4. Mr. Brian Harrison (Business Systems Controller) & Ms. Marilyn Lucas (Business Systems Analyst), Interview/Demonstration, The Automobile Association,22/4/85
1.2. Hitchman S,Relational Database Management Systems in Education, COMPUTING - The Magazine,August 29th 1985 p22
5.5. Hitchman S,Database Management Systems. COMPUTING - The Magazine,October 2nd 1986 p36
2.13. Holmes E,DBMS For Kids,DATAMATION,April 15th 985 pps. 70-78
6.6. Jenner S C,dBASEII CAL Package,Ashton Tate,1982 Version 2.0

2.2. Koffman E P,Miller P L,Wardle C E,Recommended Curriculum for CS1 1984, Communications of the ACM,Oct. 1984 pps. 998 - 1001
3.15. Lancashire G,Selecting A Database,Computing Magazine, April 18 1985 p.23
2.15. Lawson H W,The Holistic Approach To Introducing Computer Systems, Computer Education,February 1985 pps. 20-24
7.3. Professor C. D. Lewis,Personal Letter,,1985
5.8. MacCallum Stewart L,The Software Hut,Computer Bulletin, March 1985 pps 8-9.
8.1. McCrone J,Post Office is hit by DB2's shortcomings, COMPUTING,March 6th 1986 p1.
4.9. McFadden F & Hoffer J A,Database Management,Benjamin/Cummings Publishing Company,1985 pps 231 -
3.2. ibid.
5.4. ibid.
5.2. Mendelson M,Quest By The River,Educational Computing, September 1985 p 10.
2.19. Mill J,Big Bang sounds alarm on relevance of IT training, COMPUTING,April 10 1986 pps 18-19
8.4. ibid.
2.7. Mill J,Having Little Time and Not Enough Money To Teach The Teachers, COMPUTING - The Magazine,September 19, 1985, pps10-11.
5.3. Mill J,Software Budgets Take a Caning,COMPUTING, March 13 1986 pps 3 - 5
2.1. Not Attrib,CS1 Syllabus Proposals,Communications of the ACM, October 1984 pps. 1002 - 1007
8.2. Not Attrib,Experts to unravel relational database myths, Informatics, March 1986 p.49.
2.12. Not Attrib,Teaching Database Courses,Computing Newsletter Article, October 1983
2.11. Not Attrib,Using Spreadsheets,Computing Newsletter Article,May 1984.
6.5. Peltu M,Introducing Computers,NCC Publication,1983 pps.61,62,63
2.9. Peltu M,NCC Understanding Computers,NCC,1980
5.7. Piper B.,dBASE II the best seller,SOFTWARE,April 1984
2.6. RSA,Syllabus Document for Computer Science and Use of Computers,RSA,1985
7.1. Sedacca B,Database (Half-witted but much faster), Financial Times,Wednesday May 1st 1985 p.10.
2.3. Steedle L F & Sinclair K P,The Introductory Course in The Accounting Curriculum,Computer Education, June 1984 pps. 27 - 32
8.6 Southern Examining Group,GCSE Sylabus 1988 Examination,March 1988
5.6. Sweet F,What if anything is a relational database, DATAMATION,July 15 1984
8.5. Wardropper J,Board puts Comal before Logo,Computer Weekly, 13/2 1986.
6.4. Willmott G M R,The Fundamentals of Computing,Heineman,1983

# A. Appendix A - Ashton Tate Package Sources

Several sections of the thesis are quoted from the Ashton Tate software packages.

The Educational Package (which is quoted in chapter five) has a reference found in the first program in the package suite (LESSONS.PRG). The reference is :

dBASE II LESSONS RELEASE 2.0
COPYRIGHT 1982, 1984 S. C. JENNER

The HELP text was also used. The reference for this is found in the file DBASEMSG.TXT and looks like this :

HELP TEXT FILE DBASEMSG.TXT
VERSION 1.16 FOR dBASE II v2.4
Copyright 1983 Ashton-Tate and RSP, Inc.
Written by Wayne Ratliff, Jim Taylor, and Howard Dickler

## B. CAL Package Modifications

This appendix outlines the modifications made to the CAL package dBASE Educational. These modifications were made to overcome some difficulties experienced when using the package with students.

### B.1. The Operating System Modification

The modification is shown as it appears in the new TEACHI.PRG file.

```
?
?"There are a few operating system facilities that you should
                                                    become familiar"
?"with. These can be found in your computer manual, or maybe
                                                    outlined to you"
?"by by your lecturer. This is one example which you will
                                                    frequently use."
WAIT
ERASE
?"'DIR' is a command you give your computer to find out what
                                                     files you have"
?"in a particular file. When you enter the 'DIR' command you
                                                        will get a"
?"screen display of all the names of the files on your current
                                                        disc"
?"'DIR' also provides information on the size of each file and
                                                       the amount"
?"of disc space available"
?
?"You will find more information on these operating system
                                                  facilities and how"
?"they work on your computer in your computer operating manual."
WAIT
ERASE
```

## B.2. Syntax Modification

This was quite easily changed by deleting the section from the file TEACH1S.PRG. The difficult part was identifying the appropriate file to change. However, the students do have to be taught about the system. In general the best method here seems to be a verbal discussion. The system does not seem amenable to CAL implementation because of the complex nature of the syntax changing system.

## B.3. Field Definition Modification

Again, the file concerned is TEACH1.PRG. Essentially the task was one of changing the text messages to make them more understandable to the novice student. In essence the text gives more direction on two points:

This is a field definition phase and no data is entered
Data definitions are for a name, address and phone number
file and will be character information

The altered portion of the TEACH1.prg file is listed below.

```
STORE 0 TO OK
ERASE
?"Now you must decide what information you wish to store in your
                                                        file"
?"Since this is a file of information concerning friends you may"
?"decide to use three fields, for name, address, and phone number
                                                        data"
?
?"Now, you must specify the fields contained in &FILE.. Each
                                                        field"
?"specification must contain a fieldname, fieldtype (either C
                                                        for"
?"character fields, N for numeric fields, or L for logical
```

```
                                                          fields),"
?"fieldlength, and the number of decimal positions in the field
                                                            MAY be"
?"specified if it is a numeric field.  Commas separate the
                                                       fieldname,"
?"fieldtype, fieldlength, and decimal specifications."
?
?"A <RETURN> enters each field.  After you have entered the last
                                                          field,"
?"press the <RETURN> key again, and the file will be CREATEd."
?
?"By the way, whenever <RETURN> appears at the end of an entry
                                                            line,"
?"it means you should press the <RETURN> key.  Don't type the
                                                    word <RETURN>."
?
?"Here are a couple of examples of fields. A 30 position character"
?"field called NAME would be entered as follows:"
?
?"     NAME,C,30"
?
?"A 5 position numeric field called AMOUNT with 2 decimal
                                                       positions would"
?"be entered as follows:"
?
?"     AMOUNT,N,5,2"
?
?"If you have decided to store data of names, addresses and phone "
?"numbers then you will need character - C - fields.  A phone number"
?"is character information (you won't use it in arithmetic)"
WAIT
ERASE
?"OK.  Try to CREATE a file.  Make the file a name and address file"
?"that contains names and addresses of some of your friends (you do"
```

# C. Appendix C - Broadsheet dBASEII Case Study

## Broadsheet System - Outline of the Case Study

This case study is based on one designed for an in-service course for teachers with no experience of computing. The course began by introducing the students to a wordprocessor system (WORDSTAR) Together with some introductory lectures. The students would then work through the following case study, which was chosen not because it was an easy example (which it isn't) but because it reflected the kind of use to which the students would put their skills and knowledge.

## C.1. The Memo System - Introduction

This is a case study designed to take the student through a data processing problem, initially involving a wordprocessed solution but developing into an integrated database and wordprocessor system. During the course of the development of the system the student will be introduced to various data processing concepts. The case study is designed around a situation which is easily assimilated by the students involved. The material is complex, and develops into a complex model.

The system deals with the coordination of a team of lecturers teaching a Computing course, but could equally apply to any administrative team. The main aims of the system are threefold:

To keep all team members informed by issuing memos.
To issue meeting details (and agendas) to all team members.
To keep an automated marking system (the Broadsheet).

There are four phases in the development of the system, and it presupposes that the students are familiar with the WORDSTAR wordprocessing package. It is envisaged that the students are given

hands on wordprocessing experience at the very beginning of the course, and that this would be their introduction to the course. During the case study the student will be introduced to the database package 'dBASE II'. The four phases are as follows:

Phase 1 - Develop a simple 'mailshot' memo system to keep lecturers informed. (Utilizing WORDSTAR and MAILMERGE).

Phase 2 - Use the dBASE II package to keep the required input for the MAILMERGE system. This is simply a list of sessions and lecturers, together with lists of students.

Phase 3 - The files created will have 'Update' problems, and in this phase the student will use Entity Relationships and Normalization to design an efficient database system for the Broadsheet. This equates to a simplified Systems Analysis.

Phase 4 - Before implementing the system (using data supplied) a plan and time estimates are produced.

Phase 5 and 6 - A simple implementation using dBASE II with single line commands. The required files will be created and maintained. The marks will be entered using the database maintenance system. The report generators will be used for all system output. Before attempting this phase the student will have completed an introduction to databases and also the dBASE Educational package.

Phase 7 - This will be a brief examination of the effects of using a menu driven command file system to interface

with the user, and the advantages and disadvantages involved.

### C.1.1. Phase One

You are asked to design and implement the following memo circular system using only WORDSTAR. Hand in a copy of the WORDSTAR file complete with command lines (you will have to comment them), together with a sample output.

This represents the current computer system. A system of sending memos and creating meeting agendas has been implemented on an ad hoc basis. The system is quite straightforward, and uses only the facilities of the WORDSTAR package. A file called BTEC/LEC is created (in non document mode) containing the names of all course lecturers, together with a description of their module. A memo/agenda skeleton file is created (called BTEC/M) which sets up page lengths,headers and footers,and also keys in the BTEC/LEC file variables. Whenever a memo or agenda is to be created, a copy is made of the skeleton file, and the text added. The MAILMERGE facility is then used to generate a copy of the memo for each lecturer in the BTEC/LEC reference file. The system has worked very well for disseminating information.

### C.1.2. Phase One - Solution

These are the required MAILMERGE commands to key in the distribution file :

```
.DF B/TECLEC

.RV NAME,SUBJECT
```

Broadsheet Case Study Details

.pl 72

.PN 1

B/TEC COMPUTING AND INFORMATION SYSTEMS DIPLOMA.

Date : 19th April 1985

For the attention of : &NAME&
&SUBJECT&

Copies to : All course lecturers
Head of Department

The text of the memo would be inserted here.

The student will have produced a set of memos, using a file of lecturer names and subject areas (called B/TECLEC from WORDSTAR) Perhaps more importantly than developing the skill of generating the memo system using WORDSTAR, the student has an introduction to the idea of commands, and command lines collected into a file. In fact the student has an introduction to PROGRAMMING, but using a wordprocessor. This is seen as a very important part of the case study since the concept of a program has been introduced without the student realizing it !! (Unlike the usual circumstance when the student is only too well aware of it).

In fact there are all kinds of possibilities in using the WORDSTAR package - conditional commands for example. The commands introduced to the students here all all file based commands that will easily

translate into dBASE II.

### C.1.3. A Problem

The system has worked very well, until the end of the year, when it was not possible to send a memo data collection sheet to every lecturer. The problem arose because the memo consisted of a request for the final marks (coursework assessment and examination) together with a list of students. However, there were different lists of students for each course, and the memo system did not allow distinction between courses. In the end a Heath Robinson arrangement consisting of a separate memo system for each course was implemented, and proved impossible to maintain. Several lecturers did not receive the correct student lists.
This prompted an assessment of administrative computerization. This in turn led to the request for the computerization of the broadsheet system as a whole.

### C.1.4. Phase Two

As a first step it will be useful to design and implement a working system, using WORDSTAR and dBASE II, which will generate course memos.

You will need to create two files, (at least) and this without doing any serious file analysis. These will be a file of student names, together with a file of session/lecturer names. You can create and maintain the files using dBASE II and also using DBASE II you can create reference files for use by the WORDSTAR memo file from Phase One.

Implement the files and use some test date to generate the dBASE II / WORDSTAR system. Again, hand in sample output from both the packages to include file structure listings, record listings, wordprocessed samples. Finally, comment on the problems that would be encountered

in maintaining the two suggested files.

These are suggested file designs:

The Session File

STRUCTURE FOR FILE: B:SESSION .DBF
NUMBER OF RECORDS: 00020
DATE OF LAST UPDATE: 27/06/85
PRIMARY USE DATABASE

| FLD | NAME | TYPE | WIDTH | DEC |
|---|---|---|---|---|
| 001 | CODE | C | 003 | |
| 002 | MODULE | C | 003 | |
| 003 | LECTURER | C | 030 | |
| 004 | S:DESC | C | 040 | |

A typical record might be :

lC
Ml MRS. R. SMITH Module 1 - Introduction To Computing

The Student File

Broadsheet Case Study Details

STRUCTURE FOR FILE: B:REPFILE .DBF
NUMBER OF RECORDS: 00070
DATE OF LAST UPDATE: 26/05/85
PRIMARY USE DATABASE

| FLD | NAME | TYPE | WIDTH | DEC |
|---|---|---|---|---|
| 001 | LASTNAME | C | 012 | |
| 002 | FIRSTNAME | C | 012 | |
| 003 | NAME | C | 030 | |
| 004 | CODE | C | 003 | |

C.2. New System Overview

Within the Technical College, courses are coordinated by a lecturer. Part of the coordinator's job is to compile and control the marks awarded to every student in their various modules. The marks are used for report compilation and for submissions to B/TEC (the Business and Technician Education Council) so that end of year certificates can be awarded. Students who do not have appropriate marks, or who have missing marks (because they have not handed in their work) will be interviewed by the coordinator.

The current manual system is simply a large sheet of paper (called the Broadsheet), which each coordinator completes with session codes, session lecturer names, and student names. A course (full or parttime) will consist of several modules. This is a list of available modules for the Computing Course:

Module 1 - Introduction To Computing
Module 2 - (Double Module Continuation)
Module 3 - People and Communications
Module 4 - Information in Organization
Module 5 - Quantitative Methods
Module 6 - Programming Concepts
Module 7 - COBOL Concepts
Module 8 - Programming Project

Broadsheet Case Study Details

Module 12 - Small Business Systems
Module 13 - (Double Module Continuation)
Module 14 - Business Systems Project
Option Module - Microelectronics
The course are of three types :

2 year full time - leading to Diploma
2 year part time - leading to Certificate.
2 year day release - leading to Certificate.

Each course may consist of a different combination of modules in any one year. Each module has a common assessment scheme with the student receiving marks for five assessments in a year, and an end of module examination mark. These marks are weighted (60% for the examination mark and 40% for the in course assessment mark) to give a final module mark. The marks are then grades according to this scheme :

Less than 40% is a FAIL
40% or more and less than 65% is a PASS
65% or more is a CREDIT

In the manual system each session lecturer (a session is any module taught to any particular group) is responsible for completing the marks set on the broadsheet that relates to their session. This is supposed to be done as soon as any marks are available so that the coordinator can pick up any student problems. In practice session lecturers do not complete the broadsheet promptly.

A decision has been taken to automate the system using the computer, based on these evaluations:

1. Because the broadsheet is a reflection of the Computer Section within the College it has been considered that a computerized system will reflect favourably on the section, who have been asked by the Principal to give a lead in College adoption of microtechnology.

2. Compilation of reports is a lengthy process, prone to error, and

reminiscent of School Report systems. If the marks were contained in a computer database the reports with marks could be prepared accurately and quickly.

3. Marks reports could be produced at any time for the benefit of students, so that they could assess their overall performance. At the moment the Broadsheets are restricted (for security since they represent the only central record of marks).

4. Lecturers using the system would be forced to use the computing equipment and this further extends the evaluation at (2).

5. Lecturers would be forced to give timely information about marks to the coordinator since reports are being compiled regularly.

6. A small system already exists, and has proved successful, for keeping a database of session details (lecturer name) used in 'mailshot' type memo distribution for information and agendas.

The task is to design and implement the required computer based broadsheet system. The decisions taken so far reflect that a feasibility study has been undertaken (by the Head of the Computing Section) and the system is deemed to be straightforward to implement. A systems investigation has yielded the above information, together with a sample broadsheet. The broadsheet is designed like this:

Course Title : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Coordinator : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

| Student Names | Session | Session |
|---|---|---|
| | Module 1 | Module 3 |
| | Lecturer XXXXXXXXXXX | Lecturer XXXXXXXXXXX |
| | M1 M2 M3 M4 M5 EX | M1 M2 M3 M4 M5 EX |

| | | |
|---|---|---|
| XXXXXXXXXXXXX | 99 99 99 99 99 99 | 99 99 99 99 99 99 |
| XXXXXXXXXXXXX | 99 99 99 99 99 99 | 99 99 99 99 99 99 |
| XXXXXXXXXXXXX | 99 99 99 99 99 99 | 99 99 99 99 99 99 |

Fig. C-1: Manual Broadsheet System Example

There are, of course, a selection of sessions across the paper, according to the course.

C.2.1. Phase Three.

Before such a system can be implemented, it is important that the user understands exactly what they are trying to do, and also creates the most efficient system. Often such systems are created using the intuitive knowledge of the user. This is simply because a proper analysis takes a good deal of time. Systems developed in this way are often poor and usually don't work as expected. To develop efficient, reliable and easily maintained systems it is important to use a structured systems development approach. There are many to choose from, but most employ two elements - Entity Relationship determination, and Normalization. Both of these structured techniques produce an efficient file design which can be directly translated to a package like dBASE II.

C.2.1.1. Entity Relation

Produce an ENTITY RELATION diagram for the system and arrive at the required files.

C.2.1.2. Normalization

Produce a data list and derive the required files in third normal form, using the structured techniques learned.

C.2.1.3. Comparisons

Check that both structured analyses produce the same results and if not correct and reassess.

C.2.1.4. Presentation

Present the findings of the team to the rest of the group.

C.2.2. Phase Three - Solution.

In fact the problem (like most real problems) is not as easy as the text book would lead us to suspect. The broadsheet contains repeating records within repeating records, and the unwary student might choose to consider the mark set as a repeating record within each session/student relation. Whilst a viable system would work this way, the final files would be unwieldy, and it is much simpler to consider that the marks obtained by a student in any particular session are a set of marks. This is quite reasonable since the system imposes a fixed number of marks for each session.

C.2.2.1. Entity Relation Solution

The entity relation is not at all easy to arrive at, because of the difficulty of deciding what the marks are related to. Most students will opt to relate the marks to the student (which is quite reasonable) but the key to getting it right is to appreciate that the marks are in fact related to the sessions. This may only emerge after Normalization. The Entity Relationship looks like this:

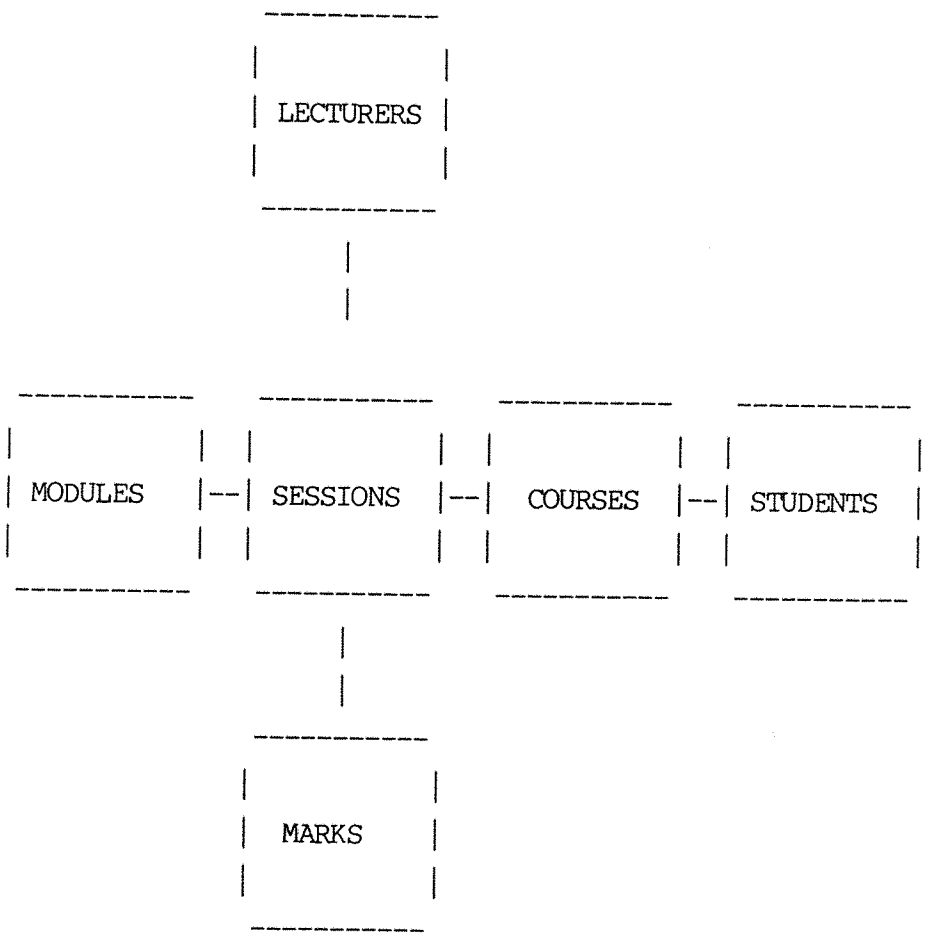
course : students 1:m

courses : modules m:m
this is split into two 1:m relations;

Broadsheet Case Study Details

| | | |
|---|---|---|
| | courses : sessions | 1:m |
| | sessions : modules | m:1 |
| marks : sessions | | m:1 |

Broadsheet Case Study Details



### C.2.2.2. Normalization Solution

The unnormalized data looks like this:

```
BROADSHEET
     COURSE:CODE COURSE:DESC
          MODULE:CODE MODULE:DESC LECTURER:NAME
               STUDENT:NAME STUDENT:DETAILS MARK:SET
               STUDENT:NAME STUDENT:DETAILS MARK:SET
               STUDENT:NAME STUDENT:DETAILS MARK:SET
               STUDENT:NAME STUDENT:DETAILS MARK:SET

          MODULE:CODE MODULE:DESC LECTURER:NAME
               STUDENT:NAME STUDENT:DETAILS MARK:SET
               STUDENT:NAME STUDENT:DETAILS MARK:SET

     COURSE:CODE COURSE:DESC
          MODULE:CODE MODULE:DESC LECTURER:NAME
               STUDENT:NAME STUDENT:DETAILS MARK:SET
               STUDENT:NAME STUDENT:DETAILS MARK:SET
               STUDENT:NAME STUDENT:DETAILS MARK:SET

          MODULE:CODE MODULE:DESC LECTURER:NAME
               STUDENT:NAME STUDENT:DETAILS MARK:SET
               STUDENT:NAME STUDENT:DETAILS MARK:SET
```

First Normal Form

```
COURSE:CODE COURSE:DESC

COURSE:CODE MODULE:CODE MODULE:DESC LECTURER:NAME

COURSE:CODE MODULE:CODE STUDENT:NAME STUDENT:DETAILS MARK:SET
```

Second Normal Form

Broadsheet Case Study Details

COURSE:CODE COURSE:DESC

COURSE:CODE MODULE:CODE LECTURER:NAME

MODULE:CODE MODULE:DESC

COURSE:CODE MODULE:CODE STUDENT:NAME STUDENT:DETAILS MARK:SET

Third Normal Form
COURSE:CODE COURSE:DESC

COURSE:CODE MODULE:CODE LECTURER:NAME

MODULE:CODE MODULE:DESC

COURSE:CODE MODULE:CODE STUDENT:NAME MARK:SET

STUDENT:NAME STUDENT:DETAILS

### C.2.3. Phase Four

You have now completed the top levels of a Systems Design, although not in the kind of detail usually (ideally) required. You are now in a position to attempt to implement your design.

For this phase you are required to plan the implementation, including giving time estimates for each stage. You may also be able to divide the tasks required amongst the team members.

### C.2.4. Phase Five

Implement the Main files and create some sample data by using the data files supplied. Decide what can be achieved simply by using dBASE II commands, including creating and maintaining all files and entering marks into the appropriate fields.

### C.2.5. Phase Six

Produce some sample reports and end-of-session reports from your system.

### C.2.6. Phase Five and Six Solution

These are the required files:

| DATABASE FILES | £ RCDS | LAST UPDATE |
|---|---|---|
| MODULE DBF | 00012 | 27/06/85 |
| REPFILE DBF | 00070 | 26/05/85 |
| MARKSHEE DBF | 00282 | 26/06/85 |
| SESSION DBF | 00020 | 27/06/85 |

STRUCTURE FOR FILE: B:MODULE .DBF
NUMBER OF RECORDS: 00012
DATE OF LAST UPDATE: 27/06/85
PRIMARY USE DATABASE

| FLD | NAME | TYPE | WIDTH | DEC |
|---|---|---|---|---|
| 001 | M:CODE | C | 003 | |
| 002 | M:DESC | C | 040 | |

Sample Output

M1 Module 1 - Introduction To Computing
M2 Module 2 - (Double Module Continuation)
M3 Module 3 - People and Communications

Broadsheet Case Study Details

M4 Module 4 - Information in Organization
M5 Module 5 - Quantitative Methods
M6 Module 6 - Programming Concepts
M7 Module 7 - COBOL Concepts
M8 Module 8 - Programming Project
M12 Module 12 - Small Business Systems
M13 Module 13 - (Double Module Continuation)
M14 Module 14 - Business Systems Project
MOP Option Module - Microelecronics

STRUCTURE FOR FILE: B:SESSION .DBF
NUMBER OF RECORDS: 00020
DATE OF LAST UPDATE: 27/06/85
PRIMARY USE DATABASE

| FLD | NAME | TYPE | WIDTH | DEC |
|---|---|---|---|---|
| 001 | CODE | C | 003 | |
| 002 | S:MODULE | C | 003 | |
| 003 | LECTURER | C | 030 | |
| 004 | S:DESC | C | 040 | |

Sample Record:

1C M1 MRS. R. SMITH Module 1 - Introduction To Computing

Broadsheet Case Study Details

STRUCTURE FOR FILE: B:REPFILE .DBF
NUMBER OF RECORDS: 00070
DATE OF LAST UPDATE: 26/05/85
PRIMARY USE DATABASE

| FLD | NAME | TYPE | WIDTH | DEC |
|---|---|---|---|---|
| 001 | LASTNAME | C | 012 | |
| 002 | FIRSTNAME | C | 012 | |
| 003 | NAME | C | 030 | |
| 004 | CODE | C | 003 | |
| 005 | HOUSE | C | 020 | |
| 006 | STREET | C | 016 | |
| 007 | TOWN | C | 016 | |
| 008 | COUNTY | C | 010 | |
| 009 | POSTCODE | C | 009 | |
| 010 | M1:GRADE | C | 006 | |
| 011 | M2:GRADE | C | 006 | |
| 012 | M3:GRADE | C | 006 | |
| 013 | M4:GRADE | C | 006 | |
| 014 | M5:GRADE | C | 006 | |
| 015 | M6:GRADE | C | 006 | |
| 016 | M7:GRADE | C | 006 | |
| 017 | M8:GRADE | C | 006 | |
| 018 | M12:GRADE | C | 006 | |
| 019 | M13:GRADE | C | 006 | |
| 020 | M14:GRADE | C | 006 | |
| 021 | MOP:GRADE | C | 006 | |
| 022 | COUR:GRADE | C | 011 | |

Broadsheet Case Study Details

STRUCTURE FOR FILE: B:MARKSHEE.DBF
NUMBER OF RECORDS: 00282
DATE OF LAST UPDATE: 26/06/85
PRIMARY USE DATABASE

| FLD | NAME | TYPE | WIDTH | DEC |
|---|---|---|---|---|
| 001 | S:NAME | C | 030 | |
| 002 | S:CODE | C | 003 | |
| 003 | MODULE | C | 003 | |
| 004 | ASSIGN | N | 002 | |
| 005 | EXAM | N | 002 | |
| 006 | AVERAGE | N | 002 | |

Sample Record

MR. J. SMITH 1D M1 63 55 0

C.2.7. Phase Seven

The student is presented with a menu driven system, like the one they have designed, but based around command files. The students can then discuss the advantages and disadvantages of such an approach. The required command files are listed below.

Because of the restrictive nature of dBASE II in some areas, it would be necessary to to make some changes to the originally specified system to accomodate dBASE II, which is rather unfortunate. The main problems encountered were:

Only being able to access two files at once.
This led to some complex 'array' handling with memory variables.

The awful effects of the BROWSE command (otherwise excellent) when handling more than a screenfull of records and the lack of a

'FOR' facility with it. This had dire effects in that the marksheet file had to be split into smaller files, one for each session - so that they fit into one screen. This could be achieved with command files for creation and updating.

The command files are presented in their raw form.

The student will probably not see the listings, and they are included for reference only. The student will see only the working system on disc. For an overview of the system look at the two menu programs called MENU and REPS.

C.2.7.1. B/TECREP

A reporting file used for a complete report on the REPFILE file based on B/TEC examination board requirements.
Title is B/TECREP

```
CLEAR
ERASE
SET TALK OFF

? "CHECK TOP OF FORM"
STORE CHR(27) + "C" + CHR(70) TO PLEN
SET PRINT ON
? PLEN
SET PRINT OFF
WAIT

SET PRINT ON
EJECT
USE REPFILE INDEX REG

? "COURSE ",CODE
```

Broadsheet Case Study Details

```
? "NAME                                M1  M2  M3  M4  M5  M6  M7  M8
                                                        M12 M13 M14 MOP"

?
STORE CODE TO T:CODE

DO WHILE .NOT. EOF

  IF CODE <> "&T:CODE"
  EJECT
    ? "COURSE ",CODE
    ? "NAME                              M1  M2  M3  M4  M5  M6  M7 M8
M8
                                                        M12 M13 M14 MOP"

    ?
    STORE CODE TO T:CODE

  ENDIF
```

```
   STORE $(M1:GRADE,1,1) + "    " TO M1
   STORE $(M2:GRADE,1,1) + "    " TO M2
   STORE $(M3:GRADE,1,1) + "    " TO M3
   STORE $(M4:GRADE,1,1) + "    " TO M4
   STORE $(M5:GRADE,1,1) + "    " TO M5
   STORE $(M6:GRADE,1,1) + "    " TO M6
   STORE $(M7:GRADE,1,1) + "    " TO M7
   STORE $(M8:GRADE,1,1) + "    " TO M8
   STORE $(M12:GRADE,1,1) + "   " TO M12
   STORE $(M13:GRADE,1,1) + "   " TO M13
   STORE $(M14:GRADE,1,1) + "   " TO M14
   STORE $(MOP:GRADE,1,1) + "   " TO MOP

  STORE NAME+M1+M2+M3+M4+M5+M6+M7+M8+M12+M13+M14+MOP TO
PRINTLINE

  ? "&PRINTLINE"
  ?

  SKIP

ENDDO

EJECT
SET PRINT OFF

CLEAR
RETURN
```

### C.2.7.2. BROADREP

This is a simple program to give a complete report on the MARKSHEET file using the report generated by dBASE II.
Title is BROADREP

```
CLEAR
ERASE
? "SET PRINTER TO TOP OF FORM "
STORE CHR(27) + "C" + CHR(70) + " " TO PLEN
SET PRINT ON
? PLEN
SET PRINT OFF
WAIT

USE MARKSHEET INDEX SES

SET PRINT ON
REPORT FORM BROADSHEET
SET PRINT OFF

CLEAR
```

### C.2.7.3. CALCULATE

This is a simple program which calculates the overall mark, based on the examination and assignment marks, and uses the necessary weightings. Title is CALCULAT.

```
CLEAR
ERASE
USE MARKSHEET

DO WHILE .NOT. EOF

  REPLACE AVERAGE WITH (0.6
EXAM) + (0.4
ASSIGN)
  SKIP

ENDDO

CLEAR
```

### C.2.7.4. END/GRAD

This is a simple program which will assess the appropriate grade comments for the B/TEC submission. Title is END/GRAD.

```
CLEAR
ERASE

USE REPFILE INDEX REG

REPLACE ALL COUR:GRADE WITH "PASS"

BROWSE FIELDS NAME,COUR:GRADE

SET PRINT ON
LIST NAME,COUR:GRADE
SET PRINT OFF

CLEAR

RETURN
```

C.2.7.5. ENTER

This program allows the entry of marks into the sub files, and using the browse command. Title is ENTER.

Allows entry of any set of marks using any of the SESSION

sheets. User can choose to get hard copies of the newmarks

```
CLEAR
ERASE
```

Main Program

Find the course,module, (ie. session) and marks field keys

```
ACCEPT "ENTER CODE (3 CHARACTERS) AND MODULE (3 CHARACTERS)" TO
KEY
IF $("&KEY",1,3) = "1DC"
  STORE $("&KEY",1,3) + $("&KEY",4,3) TO T:FILE
ELSE
  STORE $("&KEY",1,2) + $("&KEY",4,3) TO T:FILE
ENDIF
?
?
ACCEPT "ENTER MARKS FIELDS  " TO T:MARK
?
?
```

Browse command to enter the data

Broadsheet Case Study Details

```
USE &T:FILE
BROWSE FIELDS S:NAME,S:CODE,&T:MARK
STORE $("&KEY",1,3) TO T:CODE
STORE $("&KEY",4,3) TO T:MOD

ERASE

ACCEPT "HARD COPY - Y or N " TO TEMP

IF "&TEMP" = "Y"

 set form length to 70
  STORE CHR(27) + "C" + CHR(70) TO PLEN
  ? "CHECK TOP OF FORM"
  SET PRINT ON
  ? PLEN
  SET PRINT OFF
  WAIT

  SET PRINT ON
  REPORT FORM NEWMARKS
  REPORT FORM NEWMARKS
  SET PRINT OFF
ENDIF
CLEAR
RETURN
```

### C.2.7.6. GRADE

This is a simple program which will assess the required grad comments based on the overall marks. Title is GRADE.

```
ERASE
CLEAR
USE REPFILE INDEX NAME
SELECT SECONDARY
USE MARKSHEET
DO WHILE .NOT. EOF
  SELECT PRIMARY
  STORE S:CODE + S:NAME TO KEY
  STORE TRIM(MODULE) + ":GRADE" TO FIELDNAME
  FIND &KEY
  DO CASE
    CASE AVERAGE < 40
      REPLACE &FIELDNAME WITH "FAIL"
    CASE AVERAGE > 39 .AND. AVERAGE < 65
      REPLACE &FIELDNAME WITH "PASS"
    CASE AVERAGE >64
      REPLACE &FIELDNAME WITH "CREDIT"
    CASE MODULE = "M1 "
      REPLACE M2:GRADE WITH M1:GRADE
    CASE MODULE = "M12"
      REPLACE M13:GRADE WITH M12:GRADE
  ENDCASE
  SELECT SECONDARY
  SKIP
ENDDO
```

### C.2.7.7. MAKEMODS

This program generates the MARKSHEET file used in mark collection, and uses the SESSION file AND REPFILE as a references. Title is MAKEMODS.

To make the marksheet (master broadsheet) using the information

about session contained in file SESSION and about students

contained in file REPFILE

```
CLEAR
ERASE
```

Initialize memory variables with lists of sessions in

each course - for a mark set for each session

```
USE SESSION

STORE " " TO ONED:LIST
STORE " " TO TWOD:LIST
STORE " " TO ONEC:LIST
STORE " " TO TWOC:LIST
STORE " " TO ONECD:LIST

DO WHILE .NOT. EOF

  DO CASE
    CASE CODE = "1D "
```

```
      STORE "ONED:LIST" TO LIST:NAME
    CASE CODE = "2D "
      STORE "TWOD:LIST" TO LIST:NAME
    CASE CODE = "1C "
      STORE "ONEC:LIST" TO LIST:NAME
    CASE CODE = "2C "
      STORE "TWOC:LIST" TO LIST:NAME
    CASE CODE = "1CD"
      STORE "ONECD:LIST" TO LIST:NAME

  ENDCASE

  IF "&&LIST:NAME" = " "
    STORE S:MODULE TO &LIST:NAME
  ELSE
    STORE "&&LIST:NAME" + S:MODULE TO &LIST:NAME
  ENDIF

  SKIP

ENDDO

DELETE FILE IT.NDX
DELETE FILE NAME.NDX

USE REPFILE
SELECT SECONDARY
USE MARKSHEET
DELETE ALL
PACK
APPEND BLANK
SELECT PRIMARY

DO WHILE .NOT. EOF
```

Broadsheet Case Study Details

```
  SELECT SECONDARY
  STORE 1 TO T:COUNT

  DO CASE
    CASE CODE = "1D "
      STORE "ONED:LIST" TO LIST:NAME
    CASE CODE = "2D "
      STORE "TWOD:LIST" TO LIST:NAME
    CASE CODE = "1C "
      STORE "ONEC:LIST" TO LIST:NAME
    CASE CODE = "2C "
      STORE "TWOC:LIST" TO LIST:NAME
    CASE CODE = "1CD"
      STORE "ONECD:LIST" TO LIST:NAME

  ENDCASE

  DO WHILE T:COUNT < LEN("&&LIST:NAME")

    REPLACE S:NAME WITH NAME
    REPLACE S:CODE WITH CODE
    REPLACE MODULE WITH $("&&LIST:NAME",T:COUNT,3)
    APPEND BLANK
    STORE T:COUNT + 3 TO T:COUNT

  ENDDO

  SELECT PRIMARY
  SKIP

ENDDO
SELECT SECONDARY
DELETE
DISPLAY
```

```
PACK

CLEAR

USE REPFILE
INDEX ON CODE + LASTNAME + FIRSTNAME TO REG
INDEX ON CODE + NAME TO NAME

USE MARKSHEET
INDEX ON S:NAME + S:CODE + MODULE TO IT
INDEX ON S:CODE + S:MODULE + S:NAME TO SES

CLEAR

RETURN
```

### C.2.7.8. PRINTIT

This is the main report printing file, used for end of session reports. Utilizes lots of printer control codes. Title is PRINTIT.

```
Main report program for end of session reports

Initialize system

CLEAR
ERASE
SET TALK OFF

Read in module descriptions to temporary memory variables

USE MODULE
STORE M:DESC TO M1:DESC
SKIP
STORE M:DESC TO M2:DESC
SKIP
STORE M:DESC TO M3:DESC
SKIP
STORE M:DESC TO M4:DESC
SKIP
STORE M:DESC TO M5:DESC
SKIP
STORE M:DESC TO M6:DESC
SKIP
STORE M:DESC TO M7:DESC
SKIP
STORE M:DESC TO M8:DESC
```

Broadsheet Case Study Details

```
SKIP
STORE M:DESC TO M12:DESC
SKIP
STORE M:DESC TO M13:DESC
SKIP
STORE M:DESC TO M14:DESC
SKIP
STORE M:DESC TO MOP:DESC
```

Set up printer control codes

```
SET FORM LENGTH TO 70
STORE CHR(27) + "C" + CHR(70) + " " TO PLEN
```

```
SETS BOLD MODE
STORE CHR(27) + "E" + " " TO BOL
STORE CHR(27) + "F" + " " TO CBOL
```

```
SETS CONDENSED MODE
STORE  CHR(15) + " " TO CON
STORE CHR(18) + " " TO CCON
```

TO SET CONDENSED BOLD USE BOLD AFTER CONDENSED !

```
WILL SET CONDENSED/ENLARGED
STORE CHR(15) + " " + CHR(14) + " " TO CONENL
```

Broadsheet Case Study Details

```
 SHOULD CANCEL AT END OF LINE !!


 SETS ENLARGED MODE (ONLY WORKS ON THE REST OF THE LINE)
STORE CHR(27) + "W" + CHR(1) + " " TO ENL
STORE CHR(27) + "W" + CHR(48) + " " TO CENL

 WILL CANCEL AT END OF LINE !!

 TO SET ENLARGED BOLD USE BOLD AFTER ENLARGE !!


 SETS DOUBLE STRIKE
STORE CHR(27) + "G" + " " TO DOU
STORE CHR(27) + "H" + " " TO CDOU


 SETS ITALICS
STORE CHR(27) + "4" + " " TO S:ITA
STORE CHR(27) + "5" + " " TO E:ITA

 Set up printer page length and top of form

? "Page Length Set Up "
SET PRINT ON
? PLEN
SET PRINT OFF
?
? "Check Top Of Form "
?
WAIT


 Main Program loop to generate reports
```

Broadsheet Case Study Details

Each record is checked for course code and appropriate comments inserted into the report form

```
USE REPFILE INDEX REG

One or more reports ?

ACCEPT "Type CODE + LASTNAME or ALL  " TO TEMP
IF "&TEMP" = "ALL"
  STORE " .NOT. EOF" TO CONDITION
  STORE " " TO TEMP
ELSE
  FIND &TEMP
  STORE " CODE + LASTNAME = "  TO CONDITION
ENDIF

DO WHILE &CONDITION "&TEMP"

DO CASE

  CASE CODE = "1D "
    STORE "Year One Diploma" TO COURSE
    STORE "first" TO YEAR
    STORE M1:DESC + M1:GRADE TO LINE1
    STORE M2:DESC + M2:GRADE TO LINE2
    STORE M3:DESC + M3:GRADE TO LINE3
    STORE M4:DESC + M4:GRADE TO LINE4
    STORE M5:DESC + M5:GRADE TO LINE5
    STORE M6:DESC + M6:GRADE TO LINE6
    STORE " " TO LINE7
```

```
    STORE " " TO LINE8
    STORE "Mr. S. Hitchman." TO SIG1
    STORE "Course Coordinator." TO SIG2


  CASE CODE = "2D "
    STORE "Year Two Diploma" TO COURSE
    STORE "second" TO YEAR
    STORE M6:DESC + M6:GRADE TO LINE1
    STORE M7:DESC + M7:GRADE TO LINE2
    STORE M8:DESC + M8:GRADE TO LINE3
    STORE M12:DESC + M12:GRADE TO LINE4
    STORE M13:DESC + M13:GRADE TO LINE5
    STORE M14:DESC + M14:GRADE TO LINE6
    STORE MOP:DESC + MOP:GRADE TO LINE7
    STORE "Final Course Grade (Fail/Pass/Distinction) : "
                                          + COUR:GRADE TO LINE8
    STORE "Mr. G. Barrett." TO SIG1
    STORE "Head Of Section." TO SIG2


 CASE CODE = "1C " .OR. CODE = "1CD"
    STORE "Year One Certificate"TO COURSE
    STORE "first" TO YEAR
    STORE M1:DESC + M1:GRADE TO LINE1
    STORE M2:DESC + M2:GRADE TO LINE2
    STORE M3:DESC + M3:GRADE TO LINE3
    STORE " " TO LINE4
    STORE " " TO LINE5
    STORE " " TO LINE6
    STORE " " TO LINE7
    STORE " " TO LINE8
    STORE "Mrs. R. Trotman" TO SIG1
    STORE "Course Tutor." TO SIG2
```

```
  CASE CODE = "2C "
    STORE "Year Two Certificate" TO COURSE
    STORE "second" TO YEAR
    STORE M6:DESC + M6:GRADE TO LINE1
    STORE M7:DESC + M7:GRADE TO LINE2
    STORE M8:DESC + M8:GRADE TO LINE3
    STORE " "TO LINE4
    STORE " " TO LINE5
    STORE " " TO LINE6
    STORE " " TO LINE7
    STORE "Final Course Grade (Fail/Pass/Distinction) : "
                                          + COUR:GRADE TO LINE8
    STORE "Mr. G. Barrett." TO SIG1
    STORE "Head Of Section." TO SIG2

ENDCASE
SET PRINT ON
? BOL
```

Broadsheet Case Study Details

```
TEXT
BASINGSTOKE TECHNICAL COLLEGE  -  Department Of Technology
ENDTEXT
?
? ENL,"COMPUTING SECTION",CENL
?

? "&COURSE","In Computing and Information Systems"
?
?
? CBOL
? "End Of Session Report.  ",DATE()
?
? "For : ",NAME
?
? "This report shows the results in each module for the","&YEAR"
? "year of the course.  A certificate will be issued by B/TEC"
? "and this will be available for collection in the Autumn."
?
?


?"MODULE                                        GRADE"
?
? "&LINE1"
? "&LINE2"
? "&LINE3"
? "&LINE4"
? "&LINE5"
? "&LINE6"
? "&LINE7"
?
? "&LINE8"
```

Broadsheet Case Study Details

```
?
?
? CON,"The mark is a percentage based on coursework and
examination
      marks. For a PASS grade the student will need to
obtain",CCON
? CON,"a mark of 40% or above, and for a CREDIT grade a mark of
65%
                                          or more is
needed.",CCON
?

IF  $(CODE,1,1) = "1"

? CON,S:ITA,"All modules must be passed in order for a student
to
         continue into the second year of the course.",E:ITA,CCON

ELSE
?

ENDIF

?
?
?
?
?
?
?
?
?
?
?
```

```
?
?
?
? "                                                    ","&SIG1"
? "                                                    ","&SIG2"
?
?
?

? NAME
? HOUSE
? STREET
? TOWN
? COUNTY
? POSTCODE
EJECT
SKIP

ENDDO

CLEAR
EJECT
EJECT
SET PRINT OFF
```

### C.2.7.9. SES/DESC

This is designed to insert appropriate comments into the SESSION file. Strictly speaking not needed in a normalized system, but required by dBASE II due to constraints (only two files open at any one time). The program will update the SESSION file using the MODULE file and alleviates update anomalies. Title is SES/DESC.
The program will also produce the required WORDSTAR reference files used in the 'mailshot' memos.

```
INSERTS APPROPRIATE MODULE DESCRIPTIONS INTO THE SESSION FILE

USING THE MODULE HEADER FILE

CLEAR
ERASE

USE SESSION
SELECT SECONDARY
USE MODULE
SELECT PRIMARY

DO WHILE .NOT. EOF

  STORE S:MODULE TO T:CODE
  SELECT SECONDARY
  LOCATE FOR M:CODE = "&T:CODE"
  SELECT PRIMARY
  REPLACE S:DESC WITH M:DESC
  SKIP
```

```
ENDDO

CLEAR


 PRINTS OUT FILE TO WS DATA FILE AND ALSO A WS LIST FILE

USE SESSION

COPY TO SESLIST SDF DELIMITED WITH ,

SET ALTERNATE TO LECLIST
SET ALTERNATE ON
LIST OFF
SET ALTERNATE OFF

CLEAR
```

### C.2.7.10. SESSION

This program, again required due to dBASE constraints, creates the separate files required for the marks entry using the BROWSE command.

SESSION.CMD

```
CLEAR
ERASE

USE SESSION
SELECT SECONDARY
USE MARKSHEET
SELECT PRIMARY

DO WHILE .NOT. EOF

  STORE CODE TO T:CODE
  STORE S:MODULE TO T:MOD
  STORE TRIM(T:CODE)+TRIM(T:MOD) TO T:FILE

  SELECT SECONDARY
  COPY TO &T:FILE FOR S:CODE = "&T:CODE" .AND. MODULE = "&T:MOD"
  SELECT PRIMARY
  SKIP
ENDDO
SET PRINT ON
LIST FILES
SET PRINT OFF
```

C.2.7.11. START

This simple program uses data from another database (on students) to create our subsystem. This was an operational requirement.

START.CMD

THIS PROGRAM WILL INITIALIZE THE FILES NEEDED IN THE MARKS

SEQUENCE

THE ORIGINAL FILE IS 'STUDENT'

STUDENT DETAILS ARE APPENDED TO 'REPFILE' WITH SHORTENED NAME

'REPFILE' IS USED TO COMMENT GRADES AND COURSES FOR PRINTING

```
CLEAR
USE REPFILE
DELETE ALL
PACK
APPEND FROM STUDENT
CLEAR
USE REPFILE
SELECT SECONDARY
USE STUDENT
SELECT PRIMARY
DO WHILE .NOT. EOF
```

```
 STORE TITLE + $(FIRSTNAME,1,1) + '. ' + TRIM(LASTNAME) TO TEMP
 REPLACE NAME WITH "&TEMP"
 SKIP
 SELECT SECONDARY
 SKIP
 SELECT PRIMARY
ENDDO


RETURN
```

### C.2.7.12. UPDATE

This is associated to the program SESSION and updates the master marksheet file information with the sub file information.

UPDATE.CMD

```
 Update master mark sheet from the session sheets

 Choice of all sheets from session file or stated session

CLEAR
ERASE

USE SESSION
STORE " " TO T:LIST

ACCEPT "Type in session(s) or ALL    " TO T:LIST


IF T:LIST = "ALL"
    STORE " " TO T:LIST

    DO WHILE .NOT. EOF

      IF "&T:LIST" = " "
        STORE CODE + S:MODULE TO T:LIST
      ELSE
        STORE "&T:LIST" + CODE + S:MODULE TO T:LIST
      ENDIF
      SKIP

   ENDDO
```

```
ENDIF
JSE MARKSHEET INDEX IT
STORE 1 TO T:COUNT

)O WHILE T:COUNT < LEN("&T:LIST")

  STORE $("&T:LIST",T:COUNT,3) TO T:CODE
  STORE $("&T:LIST",T:COUNT + 3,3) TO T:MOD
  STORE TRIM("&T:CODE") + TRIM("&T:MOD") TO T:FILE
  SELECT SECONDARY
  USE &T:FILE
  DO WHILE .NOT. EOF
    STORE S:NAME + S:CODE + MODULE TO KEY
    STORE EXAM TO T:EXAM
    STORE ASSIGN TO T:ASSIGN
    SELECT PRIMARY
    FIND &KEY
    REPLACE EXAM WITH T:EXAM
    REPLACE ASSIGN WITH T:ASSIGN
    SELECT SECONDARY
    SKIP
  ENDDO

  STORE T:COUNT + 6 TO T:COUNT
NDDO
LEAR
```

### C.2.7.13. MENU

A simple menu program to tie up all non reporting files.

MENU.CMD

```
MAIN MENU PROGRAM FOR THE BROADSHEET SYSTEM

CLEAR
ERASE
SET DEFA TO B
@  1,1 SAY "Broadsheet Marks And Reports Package"
@  3,1 SAY "Key in your option choice : "
@  5,1 SAY "1 START      - Makes REPFILE from the STUDENT file"
@  7,1 SAY "2 MAKEMODS   - makes broadsheet and indexes using
                                                  REPFILE and SESSION"
@  8,1 SAY "3 SESSION    - creates session marksheets using the
                                                  SESSION file"
@ 10,1 SAY "4 ENTER      - allows mark entry on any SESSION file
                                                  and on any mark"
@ 11,1 SAY "               set field.  Hard copy available if
                                                  required using the"
@ 12,1 SAY "               NEWMARKS report form."
@ 14,1 SAY "5 UPDATE     - updates the broadsheet (file MARKSHEET)
                                                  from SESSION"
@ 15,1 SAY "               files, using any or all session files"
@ 17,1 SAY "6 CALCULATE - calculates the average end of session
                                                  mark into"
@ 18,1 SAY "               the master broadsheet file - MARKSHEET"
@ 20,1 SAY "7 GRADE      - replaces the grades comments fields in
                                                  REPFILE according"
@ 21,1 SAY                 to the average mark recorded on MARKSHEET"
@ 23,1 SAY "8 END/GRAD   - inserts a PASS grade into all end of
                                                  course comment"
```

```
@ 24,1 SAY "                FIELDS and allows browse to change as
                                                          required"
@ 26,1 SAY "9 SES/DESC  - inserts the module description into the
                                                          SESSION file"
?
WAIT TO TEMP

DO WHILE &TEMP <> 0

        DO CASE
                CASE &TEMP = 1
                 DO START
                CASE &TEMP = 2
                 DO MAKEMODS
                CASE &TEMP = 3
                 DO SESSION
                CASE &TEMP = 4
                 DO ENTER
                CASE &TEMP = 5
                 DO UPDATE
                CASE &TEMP = 6
                 DO CALCULATE
                CASE &TEMP = 7
                 DO GRADE
                CASE &TEMP = 8
                 DO END/GRAD
                CASE &TEMP = 9
                 DO SES/DESC
        ENDCASE
ENDDO
CLEAR
```

**C.2.7.14. REPS**

Broadsheet Case Study Details

Another simple menu program to tie in all reporting programs.

```
 REPS.CMD
 MASTER MENU FOR HARD COPY FROM THE BROADSHEET SYSTEM
CLEAR
ERASE

@ 1,1 SAY "Reports available from the Broadsheet System"
@ 3,1 SAY "Key in your choice"
@  5,1 SAY "1  PRINTIT   - end of session reports for all
                                    or any selected student"
@  8,1 SAY "2  BRODREP   - broadsheet printout of entire
                                     marks in session order"
@ 11,1 SAY "3  B/TECREP  - B/TEC report on final gradings
                                          for each course"
?
?
?
WAIT TO TEMP
DO WHILE &TEMP <> 0
       DO CASE
                CASE &TEMP = 1
                 DO PRINTIT
                CASE &TEMP = 2
                 DO BROADREP
                CASE &TEMP = 3
                 DO B/TECREP
       ENDCASE
ENDDO
```

## C.3. Summary

This rather involved documentation reflects an essentially straightforward system. Many of the complexities were caused because dBASE II was inadequate for the task, in some respects.

One conclusion from the case study is that it would be unwise to embark on the use of dBASEII with a group of students, for other than reasonably simple tasks requiring flat files, or at most two tables. Relational systems cannot be properly created in dBASEII because of the two file restriction, and this caused a great deal of complexity in programming what would otherwise have been a simple system. In the context of the group of students targetted, the aim was to demonstrate to them the final system, and not to teach them how to generate the final system themselves.

It is doubtful if this case study would prove successful when used with totally naive or casual users. The most successful part of the case study concerns the first phases which deal with mailshot memos. The later stages of student mark collection prove too complex to be fully appreciated by the case study students, who would really require 'end user' status for such a system, rather than system development status. However, the exposure of the students to the material do give them a clearer understanding of the kind of work involved in developing such a system, if not a clear understanding of how the development is implemented.

# D. Appendix D - Survey Analysis Procedures

Five files were created using dBASE II. These were:

| DATABASE | FILES | # RCDS | LAST UPDATE |
|---|---|---|---|
| CONTACTS | DBF | 00024 | 14/02/86 |
| ANALP2 | DBF | 00025 | 15/12/85 |
| PART1 | DBF | 00021 | 14/02/86 |
| PART2 | DBF | 00021 | 14/02/86 |
| ANALP1 | DBF | 00050 | 15/12/85 |

Fig. D-1: Questionnaire Files

The company listing formed the basis of the file (CONTACTS.DBF) which was used for preparing labels and mailshot type letters. Two other files were used to store the responses (PART1.DBF and PART2.DBF) and two control files were used to analyse the responses.

The analysis was split into the two sections of the questionnaire and was based around the two files (ANALP1.DBF and ANALP2.DBF) which contained all of the possible responses (from yes/no and alternative questions) available to respondents. A simple dBASE command file (ANP1.PRG and ANP2.PRG) was created for each of the files. These command files used the COUNT command to read through the files of responses and count the occurences of all of the possible responses. This was printed out to paper during the processing and a hard copy of response counts obtained.

In addition listings were obtained of the text responses. These text responses were summerized versions taken from the questionnaires.

The question counts were then input to a spreadsheet / graphics package (GRAPHPLAN) and bar charts produced for each question.

## D.1. Files Used For Data

There were three data files. CONTACTS.DBF was used to store the name and address of the contacts and firms involved. The structure of the file is shown below:

```
STRUCTURE FOR FILE:  B:CONTACTS.DBF
NUMBER OF RECORDS:   00024
DATE OF LAST UPDATE: 14/02/86
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH    DEC
001       NAME          C    025
002       POSITION      C    025
003       COMPANY       C    026
004       BUSINESS      C    040
005       ADDRESS1      C    025
006       ADDRESS2      C    025
007       ADDRESS3      C    015
008       ADDRESS4      C    015
009       ADDRESS5      C    009
010       PHONE         C    016
011       DUPLICATE     C    001
** TOTAL **              00223
```

Fig. D-2: CONTACTS File

The other two data files were used to store the responses from the questionnaires. The questionnaire was split into two part, since this would otherwise have meant that the number of fields restriction of dBASE II would have been exceeded. This restriction did not affect the analysis.

The file for part one of the questionnaire is PART1.DBF and the file structure is shown below :

```
STRUCTURE FOR FILE:   B:PART1    .DBF
NUMBER OF RECORDS:    00021
DATE OF LAST UPDATE:  14/02/86
PRIMARY USE DATABASE
FLD       NAME       TYPE  WIDTH    DEC
001       NAME        C     025
002       COMPANY     C     026
003       A1:1        N     004
004       A1:2        N     003
005       A1:3        N     002
006       A1:4:1      C     001
007       A1:4:2      C     001
008       A1:4:3      C     001
009       A1:4:4      C     001
010       A1:4:5      C     001
011       A1:4:6      C     001
012       A1:4:7      C     001
013       A1:4:8      C     001
014       A1:4:9      C     001
015       A1:4:10     C     001
016       A1:4:11     C     001
017       A1:4:12     C     001
018       A1:4:13     C     001
019       A1:4:OTHER  C     050
020       A1:4:14     C     001
021       A1:4:15     C     001
022       A1:4:16     C     001
023       A1:4:17     C     001
024       A1:4:18     C     001
025       A1:5:1      C     001
026       A1:5:2      C     001
027       A1:5:3      C     001
028       A1:6        C     050
029       A1:7:1      C     001
030       A1:7:2      C     001
031       A1:7:3      C     001
032       A1:8        C     050
** TOTAL **                00235
```

Fig. D-3: PART1 File

The other file used to store the responses from part 2 of the questionnaire is detailed below.

```
STRUCTURE FOR FILE:  B:PART2    .DBF
NUMBER OF RECORDS:   00021
DATE OF LAST UPDATE: 14/02/86
PRIMARY USE DATABASE
FLD       NAME        TYPE WIDTH   DEC
001       NAME          C    025
002       COMPANY       C    026
003       B2:1          C    001
004       B2:1:WHICH    C    050
005       B2:2          C    001
006       B2:3          C    001
007       B2:3:WHICH    C    050
008       B2:3:HELP     C    001
009       B2:4          C    001
010       B2:5          C    001
011       B2:6          C    001
012       B2:7          C    001
013       B2:7:REAS     C    050
014       B2:OTHER      C    050
015       C3:2:MORE     C    020
** TOTAL **                 00280
```

Fig. D-4: PART2 File

## D.2. Files Used For Control/Analysis

Two files data files were used for analysis, one for each part of the questionnaire. These files contained the total of all possible responses to the yes/no or multichoice questions.

The files ANALP1.DBF and ANALP2.DBF are detailed below and on the following page, together with a listing of the file contents.

```
STRUCTURE FOR FILE:  B:ANALP2   .DBF
NUMBER OF RECORDS:   00025
DATE OF LAST UPDATE: 15/12/85
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH    DEC
001      QUESTION      C     010
002      RESPONSE      C     001
** TOTAL **                00012
. LIST
00001   B2:1         Y
00002   B2:1         N
00003   B2:1         D
00004   B2:2         Y
00005   B2:2         N
00006   B2:2         D
00007   B2:3         N
00008   B2:3         U
00009   B2:3         E
00010   B2:3         D
00011   B2:3:HELP    Y
00012   B2:3:HELP    N
00013   B2:4         Y
00014   B2:4         N
00015   B2:4         D
00016   B2:5         Y
00017   B2:5         N
00018   B2:5         D
00019   B2:6         M
00020   B2:6         B
00021   B2:6         D
00022   B2:6         N
00023   B2:7         Y
00024   B2:7         N
00025   B2:7         D
```

Fig. D-5: Control File ANALP2

```
STRUCTURE FOR FILE:   B:ANALP1   .DBF
NUMBER OF RECORDS:    00050
DATE OF LAST UPDATE:  15/12/85
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH    DEC
001       QUESTION     C     010
002       RESPONSE     C     001
** TOTAL **                00012
. LIST
00001   A1:4:1      Y
00002   A1:4:1      N
00003   A1:4:2      Y
00004   A1:4:2      N
00005   A1:4:3      Y
00006   A1:4:3      N
00007   A1:4:4      Y
00008   A1:4:4      N
00009   A1:4:5      Y
00010   A1:4:5      N
00011   A1:4:6      Y
00012   A1:4:6      N
00013   A1:4:7      Y
00014   A1:4:7      N
00015   A1:4:8      Y
00016   A1:4:8      N
00017   A1:4:9      Y
00018   A1:4:9      N
00019   A1:4:10     Y
00020   A1:4:10     N
00021   A1:4:11     Y
00022   A1:4:11     N
00023   A1:4:12     Y
00024   A1:4:12     N
00025   A1:4:13     Y
00026   A1:4:13     N
00027   A1:4:14     Y
00028   A1:4:14     N
00029   A1:4:15     Y
00030   A1:4:15     N
00031   A1:4:16     Y
00032   A1:4:16     N
00033   A1:4:17     Y
00034   A1:4:17     N
00035   A1:4:18     Y
00036   A1:4:18     N
00037   A1:5:1      Y
00038   A1:5:1      N
00039   A1:5:2      Y
00040   A1:5:2      N
00041   A1:5:3      Y
00042   A1:5:3      N
00043   A1:7:1      Y
00044   A1:7:1      N
00045   A1:7:2      Y
00046   A1:7:2      N
00047   A1:7:3      Y
00048   A1:7:3      N
00049   A1:8        Y
00050   A1:8        N
```

Fig. D-6: Control File ANALP1

## D.3. Command Files Used For Analysis

Only three command files were used. Each of the question control files requires a command file. The command file simply reads through the control file and COUNTS for each question the number of responses listed in the control file.

These files are listed below.

```
CLEAR
SET TALK ON
USE ANALP1
SELECT SECONDARY
USE PART1
SELECT PRIMARY
SET PRINT ON
DO WHILE .NOT. EOF

  STORE QUESTION TO TQ
  STORE RESPONSE TO TR
  SELECT SECONDARY
  SET TALK OFF
  STORE "COUNT FOR  &TQ = '&TR' TO HOWMANY" TO WHICHANS
  SET TALK ON
  &WHICHANS
  ?
  ?
  SELECT PRIMARY
  SKIP
ENDDO
SET PRINT OFF
```

Fig. D-7: Command File ANP1

```
CLEAR

USE ANALP2
SELECT SECONDARY
USE PART2
SELECT PRIMARY
SET PRINT ON
DO WHILE .NOT. EOF

  STORE QUESTION TO TQ
  STORE RESPONSE TO TR
  SELECT SECONDARY
  SET TALK OFF
  STORE "COUNT FOR  &TQ = '&TR' TO HOWMANY" TO WHICHANS
  SET TALK ON
  &WHICHANS
  ?
  ?

  SELECT PRIMARY
SKIP
ENDDO
SET PRINT OFF
```

Fig. D-8: Command File ANP2

The final command file calls the other two and also produces the lists of the text responses to the questionnaire. Rather than producing the printed listing directly from dBASEII the command file creates a disc file listing suitable for wordprocessing. The file is detailed on the next page.

```
CLEAR
SET ALTERNATE TO WIP
SET ALTERNATE ON
? "QUESTIONNAIRE ANALYSIS  -  RESPONSE TOTALS"
?
? "ANALYSIS FOR PART 1 OF THE QUESTIONNAIRE"
?
? "REPORT OF EMPLOYEE NUMBERS BY COMPANY"
USE PART1
REPORT FORM NOS
?
DO ANP1
? "ANALYSIS FOR PART TWO (AND THREE) OF THE QUESTIONNAIRE"
?
?
DO ANP2
CLEAR
USE PART2
? "LISTING OF USERS DATABASE PACKAGES (QUESTION 2.1.2)"
?
LIST FIELDS B2:1:WHICH
?
? "LISTING OF USERS RECOMMENDED DATABASE PACKAGES (QUESTION 2.3.2)"
?
LIST FIELDS B2:3:WHICH
?
? "LISTING OF USERS REASONS FOR QUESTION 7 (DO YOU THINK USE OF DBASE II ..)"
?
LIST FIELDS B2:7,B2:7:REAS
SET ALTERNATE OFF
QUIT TO "WS"
```

Fig. D-9: Command File ANALYSIS

## D.4. Labelling / Report Command Files

Two more command files were required. The first was a report file and

was used to produce the company listing. For this purpose the file CONTACTS.DBF was indexed on the COMPANY field to an index COMP. The report file is listed below.

```
CLEAR
ERASE
SET TALK OFF
USE ADDRESS
SET PRINT ON
DO WHILE .NOT. EOF
  STORE 1 TO C
  DO WHILE C < 6
    DISPLAY NAME,POSITION
    DISPLAY COMPANY,BUSINESS
    DISPLAY ADDRESS1
    DISPLAY ADDRESS2
    DISPLAY ADDRESS3
    DISPLAY ADDRESS4
    DISPLAY ADDRESS5
    DISPLAY PHONE
    DISPLAY DUPLICATE
    ?
    SKIP
    STORE 1+C TO C
  ENDDO
  EJECT
ENDDO
SET PRINT OFF
```

Fig. D-10: Command File PRINTER

The final command file was used to generate the labels used in the mailshot system for questionnaire distribution. This is listed below.

```
* THIS PROGRAM WILL GENERATE LABELS FROM A DBASE II FILE.
* THE LAST RECORD MAY GENERATE UP TO THREE LABELS !
SET TALK OFF
CLEAR
ERASE
STORE " " TO F1
STORE " " TO F2
STORE " " TO F4
STORE " " TO F5
STORE "            " TO F6
STORE "            " TO F7
STORE "                 " TO F8

STORE "    " TO G

STORE CHR(27) + "C" + CHR(70) + " " TO PLEN
? "PAGE SET UP"
SET PRINT ON
? PLEN
SET PRINT OFF
?
? "CHECK TOP OF FORM"
?
WAIT

STORE "0" TO CHECK
DO WHILE "&CHECK" = "0"
  SET PRINT ON
  STORE "XXXXXXXXXXXXXXXXXXXXXXXXX" TO T
  STORE " " + "&T" + "&G" + "&T" + "&G" + "&T" TO T
  ? T
  ? T
  ? T
  ? T
  ? T
  ? T
  ? T
  ? T
  ?

  SET PRINT OFF

  ? "KEY IN 0 TO TRY AGAIN !"
  ?
  WAIT TO CHECK

ENDDO
```

Fig. D-11: Command File LABELS

```
USE CONTACTS INDEX COMP
SET PRINT ON
DO WHILE .NOT. EOF

  STORE " " + NAME +       "&F1" + "&G" TO L1
  STORE " " + POSITION +   "&F2" + "&G" TO L2
  STORE " " + COMPANY +            "&G" TO L3
  STORE " " + ADDRESS1 +   "&F4" + "&G" TO L4
  STORE " " + ADDRESS2 +   "&F5" + "&G" TO L5
  STORE " " + ADDRESS3 +   "&F6" + "&G" TO L6
  STORE " " + ADDRESS4 +   "&F7" + "&G" TO L7
  STORE " " + ADDRESS5 +   "&F8" + "&G" TO L8


  SKIP

  STORE "&L1" + NAME +       "&F1" + "&G" TO L1
  STORE "&L2" + POSITION +   "&F2" + "&G" TO L2
  STORE "&L3" + COMPANY +            "&G" TO L3
  STORE "&L4" + ADDRESS1 +   "&F4" + "&G" TO L4
  STORE "&L5" + ADDRESS2 +   "&F5" + "&G" TO L5
  STORE "&L6" + ADDRESS3 +   "&F6" + "&G" TO L6
  STORE "&L7" + ADDRESS4 +   "&F7" + "&G" TO L7
  STORE "&L8" + ADDRESS5 +   "&F8" + "&G" TO L8

  SKIP

  STORE "&L1" + NAME TO L1
  STORE "&L2" + POSITION TO L2
  STORE "&L3" + COMPANY TO L3
  STORE "&L4" + ADDRESS1 TO L4
  STORE "&L5" + ADDRESS2 TO L5
  STORE "&L6" + ADDRESS3 TO L6
  STORE "&L7" + ADDRESS4 TO L7
  STORE "&L8" + ADDRESS5 TO L8


  SKIP

  ? L1
  ? L2
  ? L3
  ? L4
  ? L5
  ? L6
  ? L7
  ? L8
  ?

ENDDO

SET PRINT OFF
QUIT
```

Fig. D-12: Command File LABELS - continued