THE CAMBRIDGE DATABASE SYSTEM

Dr Alan Macfarlane

in collaboration with

Dr Martin Porter
Michael Bryant

v.1.5 June 1991

Department of Social Anthropology,
University of Cambridge,
Free School Lane,
Cambridge, CB2 3RF
England

TABLE OF CONTENTS [Page numbers have altered]

THE MANUAL AND HOW TO READ IT.....6

PART A: MANUAL

CHAPTER ONE. AN OVER-VIEW OF THE SYSTEM.

Databases.....7

Files of data.....7

Records and their structure.....7

Information fields.....8
Fields within records, maximum number and length
The information part of a record field; field size
String and integer information in the fields
Group fields and their use
Sub-fields within fields

Indexing and captioning.....9

Searching and information retrieval.....10
Free text and structured queries in general
Some general features of 'probabilistic' searching
Expanding queries and making associations

Summary.....11

Size, speed and openness.....12

CHAPTER TWO. THE STRUCTURE AND CONTENT OF RECORDS.

Introduction.....13

Numbering the records.....13
Identity fields and record types
Types of identity fields for videodisc materials
Sound codes
Automatic numbering done by the computer
The automatic numbering program
Reference numbers in the original archives
Titles of the sources
The medium in a multi-media videodisc

The major part of indexing.....16
The short text, length and nature
Constraints on what to put in the short text
The longer text and its uses
Mixing the shorter and longer text fields in indexing

Keyword fields.....16
The need for keyword fields
Uses for general keywords Specific keywords for names, places, groups, dates
Uses for specific keywords
How to indicate a series of keywords
The note field
Cross-referencing
Group fields
Special features for museums, archives and libraries
A summary of the general fields
Special archival and museum fields and sub-fields
Expanding the number of fields; some spare codes
Setting up your own format

Some standardised conventions for data entry.....21
Conventions in typing dates
Conventions concerning typing of personal names
Conventions for typing river and mountain names
Ranges of non-textual items

Setting constants for repeated fields.....22

Cross-references between records.....22

The five types of record and their structure.....23
R-records
T-records
A-records
H-records
Q-records

CHAPTER THREE. CHECKING DATA AND PUTTING IT INTO A DATABASE.

Preparing, cleaning and building records.....25
Building the records as a check of format
Automatic numbering avoiding checking
The value of number checking to trap typing errors
A further stage; for information only

How to set up text files.....26

Setting up the database.....27

Two types of database system; DB and DA databases
How to set up a Database (DB) database
How to extend a Database (DB)
How to set up a Direct Access (DA) database

How to add material to a database.....30
Batchadding material
The other method of adding data to a DB database

Modifying the database.....31
Deleting single records
Editing the data from within the database
The ability to go out into other programs
Printing out materials from the database.....33
Sophisticated ways of printing out records
Printing out selected fields
Simpler ways of printing out records

Choice of indexing terms in the database.....34
The indexing program to select terms for indexing

Modifying the macros.....34

Ways of entering the system.....35
To enter 'muscat', 'muscatel', 'discatel' and CDS
To enter the DB or DA systems
Turning the record numbers on an off
Monochrome and colour screens
Turning the videodisc connection on and off
One-screen and two-screen versions
To go into the system with a marked file
The number of records to be retrieved
Selecting which database to enter
Selecting which set of macros and which database to use

CHAPTER FOUR. SEARCHING THE DATA.

Controlling the system.....39
The first choices on the introductory page
Controlling the screen and making selections
Moving back up the system

Types of query.....40
Types of searching system
A hierarchical table of contents
Free text queries
Structured queries
Combining free text and structured queries
The base page

Free text queries and how to make them.....42
How to make a free text query
The order of being shown records

Structured queries and how to make them.....43
Setting up a structured query
Finding dates; spans and years
Finding dates; days, months and years
Finding persons
Finding ethnic group
Finding locality or place

Combined free text with structured queries.....47

The ways of looking at answers to queries.....46
The three modes of looking at the material; records
Data retrieval style
Caption retrieval style
Moving between records, images and texts.....49
Moving from record to record
Seeing a piece of text
Reading texts

Expanding queries; the marking system.....50
Seeing the terms by which a record is indexed
Full relevance feedback; marking the records

Setting up files of marked items as tutorials.....53
Two forms of marking and captioning
Saving, editing and re-entering the marked files

PART B: APPENDICES

A. Technical details concerning cross-references.....55

B. Some suggestions on data entry and error corrections.....56

C. The basis of the probabilistic retrieval system.....58

D. How to make sequential searches outside the database.....59

E. A parallel system working in q.....60

F. Muscat on MS-DOS.....62

G. Some features of the indexing system.....68

H. Musquito: a text processing utility.....69

I. Files and macros.....71

J. Some examples of edited records with their codes.....79

K. Introductory choice and help pages.....84

L. How the various fields are indexed; a summary.....86

M. Cross references between records.....88

N. How to make an embedded query.....92

O. Discatel; the elementary disc cataloguing system.....95

P. Using Discatel with Muscatel.....105

Q. How to add new fields.....109

R. Full texts of the Cambridge Database System macros.....120

S. Full texts of the Discatel macros.....134

T. A quick start to CDSi.....136

U. Two ways of searching for a set of records.....138

V. The Cambridge Database System Interactive (CDSi).....139

PART C: TUTORIALS

1. Preparing records and loading them into a database.....145
2. How to find data in a database system.....153

THE MANUAL AND HOW TO READ IT

A simpler, interactive, version of the Cambridge Database System (CDS) is described in Appendix V. CDS interactive (CDSi) allows you to start on building a database and entering data straight away. It would be worth starting by trying out CDSi A 'quick start' to the system is provided in Appendix T.

The following manual provides a working introduction and reference work for the full and more powerful Cambridge Database System 2000. It is written for both those who will use the system on its own, and those who will link it to optical media, such as videodisc. Readers who do not intend to use a videodisc should therefore ignore the few sections which specifically explain how to operate CDS (the Cambridge Database System) with

optical media.

This database system is developed from the 'Muscat' (Museum Cataloguing) package written by Dr Martin Porter. Occasionally cross-references will be made to the more technical documentation provided in the Manuals to Muscat. The abbreviations used in such cross-references are:

Martin Porter, **Muscat Manual** (4th Edition, January 1990), 243pp.

Martin Porter, **Introduction to Muscat** (2nd Edition, March 1989), 160pp.

These Manuals, which are supplied with CDS, are the technical manuals which can be used to deepen your understanding of the system.

This introductory manual is divided into three main parts.

Part A deals with the system sequentially in four chapters.

Part B contains a number of technical appendices about more specific topics.

Part C consists of two tutorials, each containing a number of exercises.

You may like to read Part A, chapter one, to get an over-view of the system and then move straight on to the Tutorials, in order to get a first feeling for how the system works. Then you might read Part A, chapters 2-4, and the appendices in Part C as they are needed. In particular, you might like to read Appendix O, which describes an elementary form of the system, allowing you to create your own small database and modify the record structure, the indexing conventions, and the way the results appear on the screen and in a print-out.

Most of the examples used in the Manual and Appendices and Tutorials are taken from a specific application, the Naga Video-disc project at the University of Cambridge.

PART A: THE MANUAL

CHAPTER ONE. AN OVER-VIEW OF THE SYSTEM

DATABASES

As with most structured databases systems, the material needs to be broken up into meaningful units. This can be conceived of as follows:

System --- database 1 --- database 2 --- database 3 and so on.

That is to say, it is possible, by choosing the appropriate names for the databases, to have several different ones held in one computer, any one of which can be made active as needed.

FILES OF DATA

Each specific database may include a number of separate files. For instance, files of indexes to artefacts, films, photographs, written texts and other materials. Thus one has the structure:

Database --- file 1 --- file 2 --- file 3 and others.

These files can be added to the database one at a time, as they are ready. There is no limit to the number of files in the database. The only constraint is the over-all size of the database. The information in the files is in normal MSDOS ASCII form.

Once inside the database these files lose their identity. The database contains records, but not files. Files are just the unit by which records are added into the database.

RECORDS AND THEIR STRUCTURE

Moving down one level, each file consists of a number of records. There is no limit to the number of records in a file. Individual records can contain up to 64,000 characters. Since it is the records which tend to be shown on the computer screen, it is sensible to keep them to roughly what will fit on one or two screens, in other words a paragraph of text. Thus one has:

files --- record 1 ---- record 2 ---- record 3 and onwards.

Each record is a separate entity; it is the most important unit

in data organisation.

There are, in fact, five types of records. The R-records, are those which are indexed and are either complete in themselves, or cross-refer to images or texts. The T-records are text records, which are reached by means of an index. The A-records are 'control' records which can be used to set up the user interface, for pages of help and for other purposes, as are 'H' or Help records. 'Q' (Query) records allow the user to set up a query which the computer will run, from within other records.

INFORMATION FIELDS

Fields within records, maximum number and length.

Each record in turn consists of a number of fields. Records are likely to have information within them which appears to fall into discrete fields, often in answer to the well-known questions "Who, what, when, where, how and why". These fields are indicated by a code or tag. There can be up to 255 separate codes per record. Since the codes can be repeated and used in many combinations, in effect the number of fields is unlimited. Fields may be entered in any order. Thus we have the structure:

record --- field 1 --- field 2 --- field 3

Code and data parts of the fields.

Each of these fields in our conventions has two parts. A code part at the start is indicated by an asterisk (*), to indicate that a new field in the record is being defined. This is followed by a letter or number or combination of these, which indicates what type of field the computer is to expect. For instance, we have decided that *t means a 'text' field, while *k means a 'keyword' field.

The information part of a record field; maximum field size.

The second part of the field consists of the actual information or data. Thus '*k fishes' would indicate a keyword field with the information or text word 'fishes'. Apart from the general upper limit of 64,000 characters per record, the information in the field can be of any length, assuming that most of the words are not indexed. A constraint does emerge from the number of indexing terms which can be taken from each record.

If the record has too many index terms, it will not go into the database and an error message will be produced. It is then

necessary either to shorten the indexed fields or to change the 'blocksize' and re-make the database.

The database is set up with 'blocks' of a certain size. There are certain advantages in not increasing the block size, but also certain difficulties. One constraint is that the initial database needs to contain at least 5 empty blocks. Thus, if one had set the block size at 10k, an initial database of at least 50k would be needed. Since the blocks reside in RAM (Random Acces Memory), there may be constraints on the block size. In practice, however, there is unlikely to be a problem.

The default blocksize for CDS is currently set at 6k bytes. This will allow you to index records of a considerable length. With such a block-size, it is possible to have a record with indexed fields containing up to about 80 lines of text, each containing about 12 words, a total of 960 words. Of course, one can also put in much longer fields which are not indexed. The constraint is the number of indexing terms extracted.

With the default blocksize, a single record can contain up to about 600 index terms. It is worth remembering that:
a) not every word is indexed (the, and, to and a few other words, as well as one-letter words, are not indexed)
b) that if you decide to index a field by **both** free text and structured query, this will generate extra terms.
c) that a day/month/year date will generate three indexing terms (if indexed in date mode).

If you want to change the blocksize, this can be done by modifying the 'create' macro in the directory \muscat\macros\cds, which specifies the blocksize.

There is another practical restriction. A field which is set as a 'caption field' (to be explained later), should ideally fit on one line on the screen, that is, it should consist of up to about eight words.

As will be explained later, there are two modes of indexing, for free text and structured searching. In relation to structured inedexing, the system will refuse to add records where a field that is being indexed for structured retrieval (whether in combination with free text mode, or in structured mode alone) is too long. The maximum length is about 25 average length words. These 25 words, and the spaces between them, are, in fact, treated in structured queries as one single 'string', which will be used to make a search for exact matches.

If you try to add a file created by a word processor which accidentally has a structured (or free text and structured) field which is too long, you will be told that a structured

field is too long and you will have to shorten the field.

String and integer information in the fields.

The information within a field can consist of either strings (that is a sequence of letters of the alphabet, numbers, punctuation marks etc., which are treated as a string of characters), or as integers. Integers are numbers which can be used for numerical calculations. The input specification defines each field as one or the other. If letters of the alphabet are typed into an integer field, the computer will indicate an error.

Group fields and their use.

The normal field contains only one type of information. But it is often the case that one will be dealing with material where some of the information applies to the whole record, while there are some sub-parts which have specific information relevant to that part only.

For instance, when a sequence of photographs have been taken rapidly of a particular event, say a dance, or there are several shots of movie film made in quick succession from different angles, it is unsatisfactory to separate them entirely as different records. On the other hand, each photograph or shot may need a special description, as well as the general description for the whole sequence. This can be represented thus:
record --- shot 1 --- shot 2 --- shot 3

In this type of record, the general heading is put at the top, and this will apply to all the records. But each sub-field may also have both a specific frame number and caption.

Sub-fields within fields.

Any field may contain within itself further sub-fields, in other words fields within fields. For instance, you can deal with the fore/surname problem by defining a structure which had a general name field (*name) which contained the two sub-fields (*forename *surname). In practice we have done this extensively only in relation to the production, collection and acquisition of artefacts. Each of these fields has to contain some other information, for instance the date, person, or place of collection of the artefact.

INDEXING AND CAPTIONING

The division of the information can be made according to your

needs. One obvious way to divide information is into substantive and administrative fields.

The substantive part includes a caption, a text field, and various keyword fields. The keyword fields allow the user to record details of people, places, ethnic groups, dates, subjects and themes.

Index terms in the database are extracted from words in caption and keyword fields and can thus be searched. But index terms are not extracted from the pure text field. For this reason, important information in the text field should be identified and inserted into the keyword fields. The information does not have to be assigned to any pre-planned hierarchical ordering of categories.

Administrative information concerns the medium (photograph, artefact, book etc.), the present location and significant details of acquisition; it could also include, in a library or museum, the shelf location of the item concerned. The information in these fields also enters the index of terms and can be searched for.

The captioning of visual images, including museum objects, is necessarily a very subjective matter. Although many attempts have been made to standardise this by providing a check-list of what should be noted, none of these can provide more than a preliminary set of categories. After a large amount of testing, we have decided on a relatively simple selection as follows.

In the captions to photographs, we have broadly described what is happening, if there is action, or what the nature of the subject matter appears to be. Any particularly striking details may be noted, for instance a particularly fine piece of ornamentation. We have tested this procedure and found that several different people looking at the same photograph independently will describe it in roughly the same way. Yet there can be little doubt that people from a different culture and with different interests would describe the photographs in different ways.

The captions can only thus be a first approximation, and users will have to add further details (often contained in other fields) after searching and analysing images. Captions are necessary, however, since database searches can only be made by presenting the user with a set of relevant answers identified by their short captions.

In the case of objects we have tried to include something on the size, materials, functions, colours, motifs of each object.

But when dealing with a complex three dimensional object, it is obvious that one can only capture a little of its complex character in words. That is why we also have a picture.

Likewise in the case of moving film, a sequence lasting twenty seconds, involving several people, could generate several pages of textual description if one noted each gesture, posture, interaction, all the material objects present. We have merely simplified this in most cases to one line, for instance "group of men and boys catching fish". Again, it will be up to users to refine what can only be a preliminary index.

The same simplification is clearly necessary with texts. Often there is a paragraph which contains information on many different topics, for instance marriage payments, political alliances, economic transactions, the interrelations of chiefs and subjects. In the short caption one can merely take out what appear to be some of the more central themes.

SEARCHING AND INFORMATION RETRIEVAL

Free text and structured queries in general.

In order to find a particular record and its attached visual or textual information, there are two main methods of searching. These are 'free text' and 'structured' (Boolean) searches. The two can be combined in this system. Structured queries (of the 'and' 'or' 'not' variety) are fairly standard in databases. However, they have certain inherent weaknesses. The number of answers retrieved is usually too large or too small; users often require an expert to compose Boolean expressions of any complexity for them; the retrieved set of answers is usually not ranked in any way, and so it is necessary to inspect the entire list in the search for relevance.

The powerful feature of this system lies in the fact that it works in a way that makes it possible to inter-act with the computer. Thus it is possible to use human insight alongside computational power to improve the quality of the questions and hence the answers.

Some general features of 'probabilistic' searching and 'relevance feedback'.

For instance, one may ask a specific question, to which the best matching answer is given, then the next best answer and so on in order of declining relevance. The user is asked whether each answer is what he or she was looking for or not. Those marked as 'relevant' are then stored by the computer. The

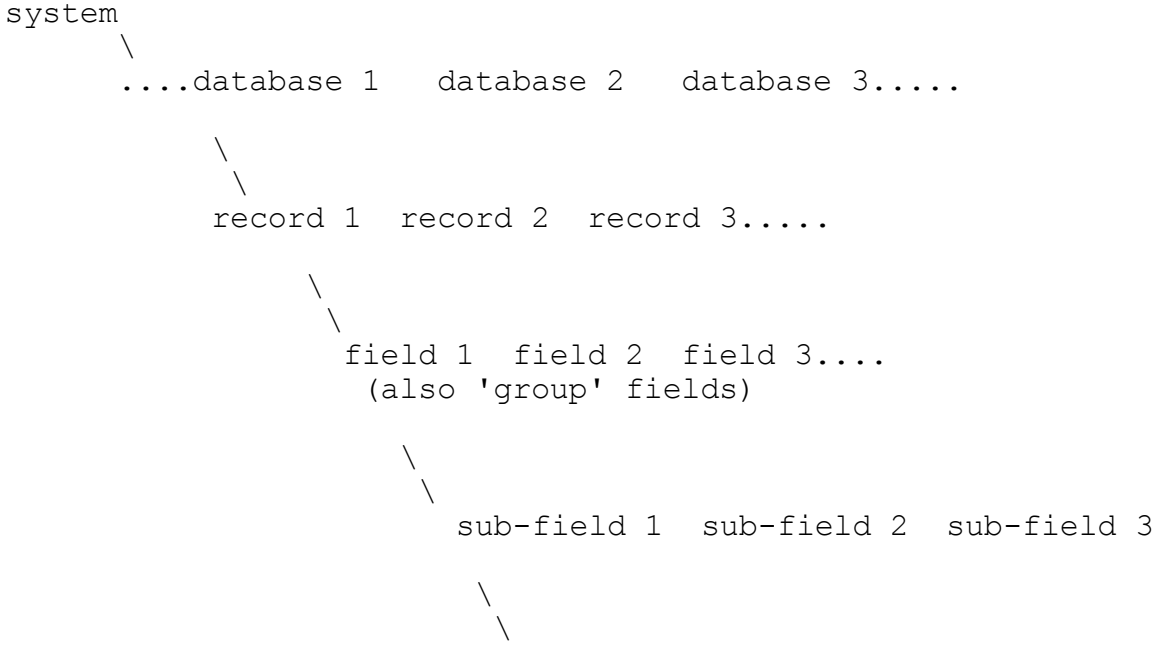
computer then presents to the user a list of the terms which appear to have been most significant in those answers marked as 'relevant'. This list will include other, associated, terms in the answers which the user may not have realised were of importance. The user is then asked to add in whichever of these new terms might be used in re-phrasing the question in a more precise form. Then a better and more powerful query is re-run, bringing out further new answers and revealing further unexpected connections.

Expanding queries and making associations in searching.

In effect the computer is helping the user to find associations which were not originally anticipated. This system is therefore a powerful tool for expanding queries and for making links between hitherto unconnected facts. The software for the system has been developed to deal with materials in museums, libraries, archives and elsewhere. It will deal with databases of any size, including visual and non-visual materials, and works on a range of desk-top microcomputers, using less than 300k of RAM within which to run.

SUMMARY

The general structure of what we have described so far may be
summarised in a diagram as follows:



code part(*) and data part

SIZE, SPEED AND OPENNESS

Thus we have a system that within one size constraint, that of a maximum of 64000 characters per record (about 15 pages of typed information on standard A4 paper) is otherwise flexible. It can be used to index most kinds of material.

It is relatively fast. Currently, searching a thirty megabyte database containing roughly four thousand pages of indexes and texts, split into some twenty thousand separate records, we find the following search times. If one asks for all the records indexed by a specific date, they are retrieved in less than two seconds; likewise all the records with a certain person or place mentioned will be found within two seconds. If one asked a structured (Boolean) query, which asked for all the records containing the intersection of a person's name, a place name and a date, the records would be found again within two or three seconds or less.

'Free text' retrieval can take longer, because the records are ranked in order of the probability of their matching the query. Thus a query with three terms, each occurring about 150 times, will produce the best hundred answers in order of likelihood in less than ten seconds. With ten terms, each with several hundred occurrences in the database, the query might take up to twenty seconds on a very slow machine, a second or two on a fast one.

The speed is increased considerably by being able to combine structured and free text searching. The machine will only take a second or two to find all the records mentioning a place name, and only a few seconds to find and order the records which match the terms in the free text part of the query. Since the system contains a sophisticated suffix-stripping or 'stemming' algorithm, it is possible to type in a word like 'marry' and get all the variants (marriage, married, marrying etc.).

The speed of retrieval is also more than doubled in 'free text' searching if the records are held in a DA (Direct Access) Database. As explained later, such a Direct Access system can be easily built from your updatable Data Base system, when needed. Structured queries in a DA database are usually answered instan-

taneously and free text queries usually within a second or two, even with thirty or more megabytes of data.

The system is an open one. The images on a videodisc may be fixed, but the indexes to them and all the subsidiary texts are held on a read/write medium (a hard disc). It is possible to delete records, change records, or continuously to add further records and texts to the database. It is also possible to extend the size of the database if it is too small. These changes can be made either from outside the database, or inter-actively from within the database itself.

CHAPTER TWO. THE STRUCTURE AND CONTENT OF RECORDS

INTRODUCTION

The following suggests a format for a structured database. It provides a template, based on a particular project, showing how you might index a set of varied materials. Of course, you may want to do this entirely differently, or extend and alter this system. A simple way of setting up your own record structure and indexing system is described in Appendix O.

NUMBERING THE RECORDS

Each record must have an unique number or identity in the database. This can either be given to the record by hand, typing it in, or generated automatically by the computer. The former method can be illustrated by looking at a particular example.

Identity fields and record types.

In the case where a record is the description of a picture on the videodisc, the identity number will also provide information about the type of material on the videodisc. This means that when material is requested, the computer can behave appropriately, showing moving film as moving, stills as still and so on.

Thus there is a code (*i) for the 'identity' field, which is followed by a letter and a number. For instance, we might have *i B.476. 'B' here stands for a photograph, and 476 is that frame number on the videodisc. Or you might have *i F.2001=2500, which would be interpreted as 500 frames of moving film, which are described in the record and played when you ask to see them.

The number sorting algorithm sorts on the first four digits of a number and is therefore inaccurate if there are less than five. The program therefore automatically adds leading zeros to videodisc frame numbers, as follows:

```
00001        -    1
00010        -   10
00100        -  100
01000        - 1000
10000        -10000
```

This is important to remember when searching for frame numbers, as explained later. Frame number 237, for example, would have to be asked for as 00237.

Types of identity fields for different videodisc materials.

We have divided our materials into the following categories, for which the following codes have been used:

Type of material	Code letter
A photograph or sequence of stills	B
Moving film	F

It is thus assumed that whenever a number is preceded by one of these capital letters in the identity (*i) field, there is a corresponding image, or set of images, available to be seen on the videodisc.

Sound codes.

Some special codes are needed for the sound. One has two sets of choices. Firstly, you can show sound and picture simultaneously (for instance if one has synchronised sound), or you can play the sound alone, without showing a picture. Secondly, you can play the sound on channel one of the videodisc, on channel two of the videodisc, or the two sounds together. This two-way and three-way choice gives rise to the following six options.

Film with audio, channel 1 = I
Film with audio, channel 2 = J
Film with audio, channels 1 and 2 together = K

Audio only, channel 1 = U
Audio only, channel 2 = V
Audio only, channel 3 = W

These prefix letters will go before the record number. For instance if you had some music on channel 2, from videodisc frame number 2000 to 2500, and wanted to turn off the picture and hear the music only, this would be:

V.2000=2500

Automatic numbering done by the computer.

The second method of numbering is done automatically. This needs to be done for textual materials in the database, which do not have corresponding images on the videodisc. For example, you might have a diary or book, each paragraph of which has been separated off as a record. Thus a book containing 300

pages, each page three paragraphs on average, might constitute a file of some 900 records. It would clearly be a waste of time to number each of these by hand.

Reference numbers in the original archives.

The original reference number of material, say the acquisition number of an object in a museum, or storage number of a photograph or document, can be preserved if necessary. This is important in order to find objects in a certain museum location, for instance. We use the field *r for this, which might contain any information that is necessary. For instance

*r Northwing 426 or *r 37.977

would be equally suitable. This is for reference purposes, and is to be found within the production, acquisition or collection fields, as described later.

Titles of the sources.

The title of the materials which is being put into the database needs to be indicated. This could be done by using the field *c . For example, you might have the title and author of a book, the name and date of a photographer. Examples of what is put in this field would be as follows:

*c photographs by J.H.Hutton taken in 1921

*c Haimendorf, 'The Konyak Nagas' (1970)

*c objects in the Pitt Rivers Museum, Oxford

*c 16mm colour film taken by Ursula Graham Bower in 1940

These titles will appear with each record, giving a viewer an idea of where material he is seeing comes from. Shown a film, photo, object or piece of text, he or she is likely to want to have a general description of what the material is and who made or collected it. This field is also indexed, so you could use it, in combination with other keywords, to find specific information within a particular source.

The medium in a multi-media videodisc.

When dealing with data in a multi-media system it is important to be able to distinguish between the different media. For instance, you might want to look at just still photographs, or just films. In order to be able to do this, you can use a *m

or 'medium' field.

*m film (moving film or video)

*m sketch (sketch, painting, drawing)

*m photograph (colour or black and white still photographs)

*m sound (audio materials)

*m artefact (three-dimensional object)

THE MAJOR PART OF THE INDEXING

The most important part of the indexing system concerns the textual descriptions. There are three levels.

The short text, length and nature.

The first level consists of a relatively short description of the record, which we call the *u or caption field. With images on the videodisc, where the description is seldom more than two or three lines long, we put this in the *u field. Basically, anything that will fit onto a small index card is likely to fit in this field.

Constraints on what to put in the short text.

The constraints on what to put in this field are as follows. This field is automatically indexed by the information retrieval system. Consequently you do not want to overload the indexes with too much material (even though a list of words like 'and' 'to' 'with' etc. are not indexed). On the other hand, if material is not put into this automatically indexed field, it is likely to be un-recoverable.

From experience with indexing museum artefacts, if a description is only a few lines long and contains important words in it, it is best to put the whole of this in the short text field.

The longer text and its uses.

The real use for the longer text field, *t , comes in longer written materials such as books or manuscripts. It is obvious that to index every word in, say, ten books, would not necessarily be a good strategy. It could be done, but would not only mean that the index would be very large, but that you would now not be able to find what you wanted because a very large number of records would come up in answer to most queries.

Mixing the shorter and longer text fields in indexing.

With paragraphs of information, therefore, the *t is used for the text, which remains unindexed. A short summary or caption of the main content of the paragraph can be provided in the short text (*u) field. For instance you might have a series of (*u) short texts for a book as follows:

*u entering the village of Wakching
*u meeting the headman and drinking tea
*u watching a man making a pot
*u boys shooting arrows; preparing for a dance
*u a man arrives with salt for trading

These would be indexed and you could thus find the paragraph by searching for headman, pot, arrows, salt and other words.

KEYWORD FIELDS

The need for keyword fields.

The short description alone may not be enough to search on, however, so you can devise a series of different kinds of keyword fields. This allows you to add in either details, or extract them from the shorter (*u) or longer (*t) fields, for special purposes.

Uses for general keywords.

The general *k keyword field includes all words that you think a user may want to search for which would not be found in the caption. For instance, you might have a paragraph of text which was mainly about marriage ritual and you had written a caption as follows:

*u the rituals used at a marriage of an old man to a young woman

But the paragraph might incidentally have some particularly striking material about other matters, so you might add:

*k opium * tigers * symbolism * archery

These will be terms that are added to the index.

Specific keywords for names, places, ethnic groups, dates.

As well as the general keywords, there are more specific ones. The principal ways of searching could be as follows:

by person name - *kp for example *kp Hutton

by locality name - *kl *kl Kohima

by ethnic group - *ke *ke Konyak

by date - *kd *kd 20.12.1939

Uses for specific keywords.

Although for some applications these specific keywords may not be necessary, to implement a full structured and free text query system it is advisable to set up some special fields. This also helps in setting up alphabetical lists of places, names and other repeated data for users. It also helps you to check for consistency and synonyms.

How to indicate a series of keywords.

If there are more than one item per keyword field, this is indicated as follows:

*kp Jones * Smith * Brown

The computer assumes that each subsequent star (*) is a new piece of information, but within the same field, in this case the *kp field.

The note field.

The above gives you almost all the fields that are needed to deal with the majority of texts, photographs, moving films and other images. It is also useful to be able to add notes of your own, which are clearly distinguished from the original text. Thus you can make comments on the material, without these becoming muddled with the original. This is done by using the *ns or 'notes' field. An example would be:

*ns this photograph was in very bad condition
*ns this has been translated from the German

This will appear separately, but alongside the rest of the text.

Cross-referencing.

A simple cross-reference can be achieved with another field, namely *qv. For example you might have:

*qv Austen, Pride and Prejudice, p.75

which would alert the user to relevant information on that page of the named book. By the insertion of a specific record number a user can be taken straight to the relevant page of text.

Group fields.

It is sometimes desirable to have items of information grouped together within a record, so that some of the material refers to all the parts, and some only to specific parts. One way to do this is through the 'group field' (*g) (See MM,p.8 for a more general description). Let us illustrate this with an example.

Supposing we had a set of photographs taken on a certain day by a photographer, all roughly concerning the same activity. We thus want to group them together in one record. On the other hand each photograph deserves a separate caption. To take a non-anthropological example, this might lead us to create the following record:

```
*kd 12.10.1986
*kl Cambridge
*u a rowing regatta; a race between Cambridge and Ely
*g *i B.2000 *u getting the boats into the water
*g *i B.2001 *u the coxes exchange a joke
*g *i B.2002 *u the end of the first section
*g *i B.2003 *u Ely cross the finishing line
*g *i B.2004 *u a friendly drink after the race
#
```

In this case the date, place and general caption would apply to all the subsections, but the material in the parts of the group would each be kept with separate headings.

Thus *g can be followed by most of the other fields, which will then be treated as referring only to that part of the record.

An alternative way to create a record with a set of linked photographs or films avoiding the use of the *g field uses the facility of repeated fields. You can simply put in the set of references you want to make, separated by a star (*), to show a repeated field.

For instance, you might have a set of still photographs and films which were numbered as follows: B.200, B.300=350, B.176, F.2000=2100, B.276.
Thus there are three stills, a series of still frames, and some

moving film, all related to the same topic. This can be put in a record as follows:

*i B.200 * B.300=350 * B.176 * F.2000=2100 * B.276.

The effect of this is that when you ask to see the pictures associated with this record, you will be taken to the first of the above, which has a 'Next' box on the menu. You can thus move very quickly through the various frames, without having to go backwards and forwards to boxes on the record.

Special features for museums, archives and library documentation.

The programs behind the Cambridge Database System were originally developed for museum use. After considerable trial and error and consultation of the appropriate museum documentation database literature we devised some extra fields. These provide the information which is often given on museum documentation cards and required by museums and archives. Although this is a specific application, it may be useful for those using the system in a museum or archival setting.

You usually need the size or measurements of an object. This is contained in the *z field, for instance:

*z 6x5 cms or *z height 7 inches, width 3 inches

This can be typed in exactly as it is needed since it is not indexed in itself.

You often need information about the provenance or origins of objects or archives. This can be broken down into who produces or made the object (*prod); who collected it from the field (*coll); and who acquired it (the museum or archive or private individual (*acq). Each of these fields has sub-fields which specify features in more detail:

*f form or method (e.g. by purchase, gift or whatever)

*p the person or persons involved

*d the date

*l the location

*e the ethnic group

*n notes

*r the reference number in the archive or museum

An example of a record of an artefact might thus be:

*u a drinking vessel made of wood
*prod *e Konyak Naga
*coll *p Butler/ Major J *acq *f gift * Butler/ Capt J *r 29.110
*n this is a son of the collector

The above roughly replicates catalogue cards. It should be noted that it is time-consuming to type all this accurately into a computer, though there are methods to speed this up, as described below.

A summary of the general fields.

Code	Nature	Notes
*i	unique number	refers to image number on disc
*c	title	in full, eg. author and title
*m	medium	for example, film or sound
*u	short caption	a short, indexed, description
*t	full text	usually a paragraph or so
*k	keyword	extra keywords can be added
*kp	person(s)	
*kl	location(s)	
*ke	ethnic group(s)	
*ns	notes	
*qv	see also	
*z	size	
*g	group field	this can contain extra information which only refers to a part; it can therefore contain all of the above codes within itself

Special archival or museum fields and sub-fields.

*prod producer or maker of the object, photograph etc.

*coll collector of object

*acq acquirer of the object

These codes must contain at least one or more of:

*f form or method

*p person or persons

*d date
*e ethnic group

*l location

*r archival or museum record number

*n note

A more technical description of how each of these fields is indexed is contained in Appendix L. If you want to try to try to add in some further fields, an example of how this is done is given in Appendix Q.

Setting up your own format.

You may want to set up your own format, using different codes with different meanings. A simple way of doing this is described in Appendix O, or, more simply, in Appendix V.

SOME STANDARDISED CONVENTIONS FOR DATA ENTRY

Conventions in typing dates.

We have standardised dates as day, month, year. If the day or month and day are missing, they are ignored. Thus we could have: 12.4.1939 or 4.1939 or 1939

A span of dates is given in the form:

24.4.1939-28.4.1939 or 4.1939-5.1940

as appropriate.

These are the standards in the date fields (*kd and *d), which are indexed. In the text fields (*u and *t), the dates can appear in any form you like, for instance May 1940, or Spring

1940.

Conventions concerning typing of personal names.

If personal names appear in the text fields, they can be in any form. But if they are in the person name fields (*kp or *p), where they will be used for indexing, they are put in the form:

Woodthorpe/ Col.R.G. or Hutton/ John

This enables you to print out alphabetical lists sorted first by surname and then by initials, titles or forename second.

Conventions for typing river and mountain names.

In order to distinguish the names of mountains and rivers from other place names, we type

Japvo Mt. meaning Japvo mountain
Zulo R. meaning Zulo river

Other abbreviations for other natural features could be developed as needed.

Ranges of non-textual items.

It is possible to specify ranges. This is done by giving the start and end number, separated by "=" or an equals sign. Thus you could have the following example:

B.100=108 - meaning photographs 100 to 108

or F.2500=2800 - film between these frame numbers.

SETTING CONSTANTS OR REPEATED FIELDS

It is often the case that a set of records will all have a field in common. It is clearly a waste of time to type the name of a photographer or the title of a book hundreds of times, each time it appears in the separate records.

There is therefore a mechanism for setting a 'constant' which the computer will automatically place with each record until this constant is cancelled or superseded. To type a constant you type in a code and date, for example *kl Kohima, and then set this as a constant by typing #m, followed by a letter. For

instance:

```
*kl Kohima  #m  a
*kd 1940    #m  b
```

would set two constants, a and b, which would be added automatically to all subsequent records until cancelled.

At the end of setting a list of constants one 'locates' them by typing #L and then typing their letters. The example above would thus look as follows:

```
*kl Kohima #m a
*kd 1940 #m b
#L a b
```

This would then be followed by the individual records, each of which would automatically have these two fields added to it, until you alter the constants.

To give a more elaborate example, you might have at the start of a file the following list of constants:

```
*c a manuscript diary by J.P.Mills #m a
*m photograph #m b
*kl Assam #m c
*ke Angami Nagas #m d
*kd 12.1925 #m e
*kp Mills/J.P #m f #L a b c d e f
```

This would then be added to each diary reference. After a while the date might change. You could then type:

```
*kd 1.1926 #m e
#L a b c d e f
```

and the new date would be inserted.

In order to suspend all the constants you type an 'empty' location (#L) list between records, as follows:

```
#L
new record......
```

To cancel all the constants, you set them, but do not give them any definition or value, as follows:

```
#M a b c d e f

new record....
```

You can now start again.

Cross-references between records and embedded queries

The preceding description gives you enough information to set up normal 'R' records. Two additional and powerful features of the system are for more advanced users, namely the ability to set cross-references between records which take you automatically to another record, and the possibility of writing a query which is 'embedded' in a record. Appendix M describes how to set up cross-references, and Appendix N explains embedded queries.

THE FIVE TYPES OF RECORD AND THEIR STRUCTURE

There are five main types of 'record' in the system. These are treated differently by CDS 2000, according to the value of the letter part of the identity.

R-records.

The R-records, which take the form, for instance, R.459, are those which are indexed within the database. When you are running an information retrieval query, this will retrieve R-records and only R-records. Only terms in the R-records are put into the index. Thus an R-record is a complete item of information, or it can cross-refer to another record. (An 'R' record can contain any number of \...\ cross-references, or a single |...| cross-reference. /..../ cross-references should not be used, and will be treated as if they were \....\ cross-references, as explained in Appendix M.)

T-records.

T-records are pages or paragraphs of text, for instance from a book or manuscript. Cross-references to other pages of text can be made from a T-record, as explained above.

A-records.

These are 'control' records, which include 'help' text pages and pages for constructing a structured ('Boolean') query. (They should not contain |...| cross-references, but are otherwise somewhat similar to T-records.)

Q-records.

These are 'query' records, which allow you to 'embed' a query within a record of any of the above types. Their purpose and nature is explained in Appendix N.

H-records.

These are 'help' records, which can be used in the introductory pages to give specific guidance to a user of the system.

CHAPTER THREE. **CHECKING DATA AND PUTTING IT INTO A DATABASE**

PREPARING, CLEANING AND BUILDING RECORDS

If you now have a file containing a set of records in the appropriate format, you will need to check their accuracy, number them, and mount them in the database. It may also be helpful to print out selected fields and to sort them to help with further indexing and checking. This chapter will explain these stages. As you will see, almost all the commands are pre-fixed by c-, for instance c-build. This indicates that a special set of 'macros' or programs are being used.

Building the records; a check of format.

All records have to be 'built' before they can be put into the database. Since a record will not 'build' unless it is in the right form, the build program is also a crucial test for accuracy in specifying the codes.

'BUILD' is the program which takes a text file of ordinary typed information, plus the codes explained above, and then converts it into a series of 'built' records, suitable for input into a database. After entering 'muscat' the simple form of the command is:

```
c-build from DATAFILENAME.txt  to NEWFILENAME
```

DATAFILENAME is your text file, which must have an extension to it of .txt, and it is copied to NEWFILENAME, which is given the extension .mus (for muscat file). An exercise showing precisely how to do this is included below in Tutorial 1. This will make use of the 'build' specification. (The creation of the build specification is described in MM,pp.34ff, and MI(Muscat Introduction),pp.71ff. If 'build' fails, errors will be listed and can then be corrected. Some suggestions on how to correct such errors is contained in Appendix B.

Automatic numbering avoiding checking.

All records have to be given a number before they enter the database. This number is most simply given by an automatic numbering program, which thereby avoids the problem of inconsistent numbering. An exercise in Tutorial 1 will take you through this. In essence, you use:

```
c-number FILENAME.MUS TO FILENAME2
```

Filename.mus is your 'built' file, which is then numbered and copied to a new name, also a built file. You will be asked what type of number is to be given (T. or R.) and what number to start at. The rest is done automatically. You need to ensure that numbers you assign do not overlap. If you give two files as starting at T.1000, for instance, the latter set of records will over-write the former.

The value of number checking to trap typing errors.

Although you may finally number the records automatically, the CHECKID program is useful to check identity numbers, that is the videodisc *i numbers. When you have long lists of numbers, for instance a thousand photographs, each with a consecutive number, it is difficult to check these by eye. CHECKID will automatically show up any errors which are reflected in a wrong order, for instance if you have typed:

*i R.40145
*i R.40146
*i R.41147
*i R.40148

and taking the built file containing these records type:

c-checkid from FILENAME

You will get the following error message:

Input recs3 and 4 not sequential.
Faulty recs (if any) left in $work\f3

You then turn these faulty records into an ordinary file by going:

c-list $work\f3 to filename

Then you can look at the file 'filename', which is in the \mus-cat\cds directory, using a word processor. Of course, you may have intended the numbers to be out of sequence, and intend to make the records acceptable to the database by covering over this fact by automatically numbering the records. In this case, these are not errors, and can be left out of sequence.

A further stage; for information only.

A further step, which the user will not see and need not worry about, is worth mentioning for information. Particular

print characters are needed to make the material appear correctly on the screen of a desk-top computer. To do this, a file is put through a special program called LISTE1. This is now built into the BATCHADD program and so is done routinely, without the user being aware of it.

HOW TO SET UP TEXT FILES

As explained earlier, only R-records are indexed, and thus only R-records will be found through an information retrieval query. Yet you may want to include very large text files, for instance books or manuscripts. You want to find a page of text, and then be able to read it and move backwards and forwards through the text.

A paragraph of a book or diary can be conceived of as consisting, in fact, of two records. There is a short caption or description of the paragraph, for instance:

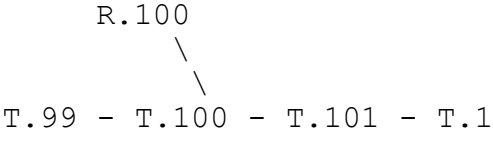
A trip to Kohima to see the governor

There may also be certain keywords, that is names of people, places, ethnic groups or whatever, which are in the paragraph or related to it, which have been abstracted and put into keyword fields. All this will go into an indexed R-record.

Such an R-Record is likely to have a title (the name of the book or description of the manuscript), a variety of names, dates, places, ethnic groups, as appropriate, and a cross-reference to the full text to which it refers. This cross-reference is of the {|T.100|} variety, mentioned above. Thus when you find the Record referring to a paragraph, you will be offered the menu option 'Show', which, when selected, will take you to the appropriate paragraph of text or T-record. From this you can move back to the R-record, or backwards and forwards through the text, paragraph by paragraph, even though these paragraphs no longer correspond to your original query.

Meanwhile the text file consists of a series of T-records, each of which has cross-references to the previous and next pages, and also, perhaps, to a table of contents.

This can be visualised thus:



Clearly, to add in all these cross-references between

records and texts, and between paragraph and paragraph of text would be a very long and tedious business by hand. There are some macros to help to do this. Exercise 5 in Tutorial 1 explains how this is to be done.

SETTING UP THE DATABASE

The cleaning of data is common to both database systems, but at this point the path diverges and it is necessary to explain the two types of database.

Two types of database system; database (DB) and direct access (DA).

Each type of database has its advantages. DB systems are more flexible in that you can add new files to the database, or delete, add or change single records, without destroying the whole database. But searches are slower than in Direct Access systems (the difference may be anything from a fraction of a second to a few seconds). Likewise, the records are not as tightly packed in the database (DB) system; the DA system may take as little as half the space to store a set of data.

How to set up a Database (DB) system.

In order to set up a DB (Data Base) system, you merely type:

```
c-create
```

You are requested to specify the number of kilobytes (1000 bytes). Thus if you want a two hundred k(ilobyte) database, you type '200' in answer to the request. This will create an empty 200k database. This database is contained in a file called DB.DA, and resides in the \muscat\cds directory, or whatever sub-directory you are in.

How to extend a Database (DB).

If, at a later point, the database fills up, you can make it larger by extending it, using the following command within Muscat:

```
dbextend  DB.DA  to DB1.DA   bytes 20000000
```

This would take the old database, db.da, and extend it by 20 megabytes, saving the material that is already in it. Thus, if you have a 10 mb database and want to extend it to 25 mb, you will extend it by 15 mb. Temporarily, you will need a total of

35 mb. while the operation is under way to accommodate both versions.

If the extension fails for some reason during the process, the old version will not be corrupted. If the extension has been successful, the old version can then be deleted. The best way to do this is to use the following MS-DOS commands:

```
rename db.da olddb.da
rename db1.da db.da
```

make sure that the new database is all right, then type:

```
del olddb.da
```

In practice, to extend a database of 10 megabytes to one of some 25 megabytes only takes three or four minutes. All this should be done from within 'muscat'.

How to set up a Direct Access (DA) Database.

There are two methods to set up a Direct Access (DA) system (which is more compact, faster, and will prevent people from changing it). One is to create it directly from a file, the second is to download it from a DB database.

In the former, all the records that are to be put into a database must be in one file. No further records can be added once the DA database is set up. At the start of the file you will need to have screen control records, as in the A.1, A.2 etc. records printed in Appendix K. A set of these is provided in the intro.txt file in the \muscat\cds directory. These may be edited to make them appropriate for your use, as indicated in Appendix K. They should then be added to the start of the file you wish to turn into a D.A. database.

When you have a clean file ready, with the screen records at the front, 'build' it, 'number' it (answering 'A' to the type of record request; starting at the number 1). Then type:

```
c-setupir  filename   (the name of your file)
```

You will now have a DA database, contained in two files called DATERM and DAREC in your current directory. Remember that doing this will over-write any other DATERM and DAREC files in that directory, which need to be re-named if you want to save them.

The other method is for use if you have already built up a DB database and then want to make a DA version. You use the command:

c-reload

This asks you , with the prompt db=, for the name of the database which you want to unload. To this you would reply db.da. This would take the file db.da (a database in fact) in your directory \muscat\cds and make a copy of it as a DA database (retaining also your original DB version).

This then does the following automatically:

```
c-unloadt   <db>  to $work\f1
c-loadt  $work\f1  to daterm
c-unloadr  <db> to $work\f2
c-loadr $work\f2 to darec
```

In plainer language, this uses four macros, which do the following.

'unloadt' takes a named database and unloads the terms in it to a temporary file called f1 in the directory \muscat\work ($ is always an abbreviation for \muscat).

'loadt' then takes this temporary file and converts it into a permanent file of the indexing terms, called 'daterm'.

'unloadr' takes the same database and unloads the records to a temporary file called f2 in the same directory, \muscat\work.

'loadr' takes the temporary file f2 and loads it into a permanent file of records called 'darec'.

If you have plenty of space, this whole procedure can be done in one go, using c-reload, as stated above. It takes some time, up to an hour for a database of over 15 megabytes on a fairly slow machine.

It should be remembered, however, that it also, temporarily, takes up a lot of space. In the final version, the DA file will probably be about half the size of the DB file. But the temporary files that are created are as big as the final DA files.

In effect, this would mean that if you started with a 10 megabyte DB file which you wanted to unload, you would need roughly the same space again, temporarily, while this process was going on, 5 megabytes for the final version of the DA file, 5 megabytes temporarily.

You can save some space by doing the process in stages. In other

words, instead of using c-reload, do each of the stages, as specified above, starting with c-unloadt. After setting up the 'daterm' file, the temporary file f1 can be deleted (with any other temporary files which tend to linger in the sub-directory \muscat\work\ and need periodic cleaning out). But the records file tends to be much the larger, so you may still have problems.

One solution, of course, is that once you have created the temporary work files f1 and f2, you can store your database somewhere else temporarily. Using the 'Backup' and 'Restore' archiving programs on your micro, you can release this space for a while, and then retrieve the DB database at the end, when you have deleted the temporary files f1 and f2, once the DA system is fully set up.

The final product consists of two files, called DAREC.DA and DATERM.DA which are held in the \muscat\cds directory.

Whatever you do, remember to remove the temporary files at the end, otherwise you will be filling your machine with a great deal of unnecessary material.

HOW TO ADD MATERIAL TO A DATA BASE (DB).

Batchadding material.

There are two main ways to add material to a DB. One of them is used for R-records, for instance indexes of photographs or artefacts or books. This is by using the program BATCHADD. What this in effect does is as follows:

```
c-build file1 to file2
c-liste1 file2 to file3
c-build file3 to file4
c-number file4 to file5
c-add file5
```

But in order to invoke it, all you have to do is to type:

```
c-batchadd
```

You will then be asked what is the name of the file from which the records are to be taken, at which, for instance, you could type millsbw.txt. Note that these are text files, that is to say 'unbuilt' files, with the extension '.txt'.

Then you are asked what type of records they are, to which you would normally reply R (capital R for R-records). Next you are

asked what number to start numbering at. Here it is essential that there is no overlap with other records of the same type. Although there is no problem if you have a set of records T.1000 onwards and R.1000 onwards, if you put in two sets of R.1000 onwards records, the set that are put in later will overwrite and destroy the earlier set.

When you have answered these prompts, a carriage return will automatically feed the records into the database. A record of how the program is proceeding will be given, with error messages if the program fails.

Depending on the number of terms to be indexed and the power of your computer, it will take between two seconds and two minutes for 100 records to go through this program. So a file with 1000 records will take between twenty seconds and twenty minutes to process. A big file of photographs, for instance a set of some 2,500 photographs, may thus take between twenty-five minutes and an hour. Since text files (that is T. records), do not themselves contain indexing terms, they can be added to the database very much more quickly.

The other method of adding data to a DB database.

If you want to add data which has already been numbered, you can use a simpler program thus:

```
c-add from  FILENAME
```

This file will then be added to the Database. This is one way to replace defective records, or to add text files. The file must be a built and numbered file, with the extension .mus, residing in your \muscat\cds directory.

MODIFYING THE DATABASE

Deleting records.

One way to modify a database is to delete faulty records. In order to delete a single record from outside the database, you can go into muscat and type.

```
c-del  R.100     (where R.100 is the record to be deleted)
```

If you want to delete a whole set of records which have been added to a database, go into the database and find the record number of the first record in the file. Supposing this is record number 500, you then do the following.

Make sure that the file of records which you have added in error is available and has the ending .txt (for instance photos.txt). Go into CDS and type:

c-batchadd photos

You will then be asked the letter code. Normally you would reply with a capital R, though it might be another letter if you have numbered the records as A, T or other records. Then you will be asked the starting number, to which you would here reply 500. All the records in that file will then be deleted from your current database.

Editing the data from within the database.

If you want to change a record or add a single record while you are in the database, this is possible by using an in-built screen editor.

You find a record you would like to change and press the control key and function key 1 (^F1).

You will now see the record displayed in its full form. It can be edited with the screen editor. This screen editor has some automatic commands, as follows:

^u - sets upper case ^l - sets lower case
^s - adds a space
^backward arrow - goes to start of a line
^forward arrow - goes to end of a line

Control with the function keys, has the following effect:

F1 - delete to end of line
F2 - delete line
F3 - go to top of file
F4 - go to bottom of file
F5 - insert line
F6 - join
F7 - delete block
F8 - windup and save

No shift (lower case) with function keys:

F1 - delete
F2 - blank line
F3 - previous line
F4 - next line
F5 - start block

F6 - end block
F7 - move block
F8 - copy block
F9 - quit without saving

You can also use the usual arrow keys, delete and other buttons to move around the text and delete or add in words.

When you have finished your editing, you can quit without saving by typing F9, as above. If you want to save the edited version, type control F8. If you windup and save the record, it is re-built and added back in its changed form into the database, all the terms being added into the index etc. It will not at first appear any different on the screen. But if you leave the record and do a new query which brings you back to it, you will see the new version that you have edited.

If you alter the record so that it contains coding errors, it will not 'build' and hence will not be added back into the database. The previous record which you tried to edit will not be altered or corrupted.

An important point to remember is that the editor described above is designed for use with an updatable, DB, system. If you are working with a compressed, DA, system, then you should not use this. The only situation in which you might want to do so is if you just wanted to see what the original record looked like, by pressing Control and F1. Then use F9 alone, so that the system does not try to put the record back into the database. If you try to put the record back, it may try to write it into any other DB file you happen to have in the current directory, which may not be what you want to do. So be careful.

The ability to go out into other programs from the database.

A similar system allows you to re-enter the 'muscat' system from within the database. You can type control F8 and you go back temporarily into Muscat (while remaining within the CDS 2000 database). You will now receive a prompt discat>, to show you that you are within muscat, but have arrived via the database. A 'stop' here will take you back to CDS 2000.

Now that you are in Muscat, you can do all the normal muscat commands, for instance you can build files, or go into the 'q' or query system. Since it is also possible to go into DOS or MSDOS from within Muscat by typing DOS or MSDOS followed by the Dos command, you could now go out into other programs. In effect, this means that you can have complete flexibility, using the Database as a shell, within which you can do anything else which your computer is capable of.

PRINTING OUT MATERIALS FROM THE DATABASE

Sophisticated ways of printing out records.

As we have seen, the text files are 'built' into a database. They cannot be read in this form, and you therefore need to unload them again in order to read them. This is done by printing them. Since the original system was designed for cataloguing, a good deal of attention has been devoted to creating sophisticated ways of printing out the records. These are described in detail in M.M. pp.72ff.

As a start you can use standard print specifications which will produce reasonably arranged records, placing the fields described above in various positions either on a mainframe terminal, on a microcomputer screen, or as print-out. The way in which the record is to be printed on the screen in CDS is declared by a print specification (pspec).

Printing out selected fields.

For special purposes you may not want to have all the fields printed out. For instance, you may want an index to the places, that is of the *kl field, but not need all the other material. By using the c-kill program (as explained in MM.p.218), it is possible to leave only the fields that are to be printed out. Some special purpose programs to do this are listed in Appendix I.

Simpler ways of printing out records.

There are two relatively simple ways to print out particular records. The simplest, apart from using your 'print screen' facility, is to use the control key combined with function key 7 (F7). This will print out on a printer the currently displayed record.

You can also 'mark' a set of records (which can be either/both R and T records) and save these by outputting them to a permanent file, as explained later. Then go into the muscat system (either from within the database or outside it). Let us say that you now have a marked file called sample.mks. You then go (in muscat),

c-getmrecs sample to sample

(or use c-getdiscm if you are accessing a Direct Access Database, that is a DA system)

c-print sample to sample1 (you can also use c-list to get

another version of the record, if you want)

You will now have an ordinary text file, called sample1.txt. This can be printed out, either by using a word-processor or, within muscat, by going:

msdos (or dos) print sample1.txt

An even simpler way of printing is described in Appendix V.

CHOICE OF INDEXING TERMS IN THE DATABASE

The indexing program to select terms for indexing.

The computer has to be told how to extract terms from the records. This is done by the INDEX program. (This is fully described in M.M.pp.154ff).

The macro contains a number of words like 'about', 'after', 'again', which are not indexed, some 80 words in all. Words consisting of only one letter are not indexed because it has been found that, in general, they are low in information content.

All the fields are indexed, except the longer text (*t) field. In the macro the line of the program which says "not find *is s eq 'T'" makes sure that this field is not indexed. If this were changed, all of the record would be indexed.

MODIFYING THE MACROS

If you want to modify the macros, to add new fields, to change the way the indexing works, or in any other way, please see Appendices I, L and P.

WAYS OF ENTERING THE SYSTEM

It is possible to enter the system in various different ways and at different levels. These allow the user to set up some of the parameters and defaults.

To enter 'muscat', 'muscatel', 'discatel' and CDS 2000.

It is possible to enter 'raw' muscat by typing 'muscat' from any directory. To enter elementary muscat (muscatel), type 'muscat sys el' from any directory.

To enter 'discatel' (elementary discat, as described in Appendix O), go into the 'Delbase' directory and type:

muscat discatel

To enter the Cambridge Database System (CDS 2000), where a number of programs/macros are already in place, type:

muscat cds

This takes you into the Cambridge Database System, and it is then possible to enter the database itself in various ways. (If you have a 'genlocked' system with a videodisc, you may want to set this up as a way of entering the system. Separate instructions for this will come with your 'genlocking' device.)

You will now receive the normal muscat> prompt. In this you can build and add files for the database, and do all the other c-commands described above

To leave any of these versions of muscat or muscatel, type 'stop'. Remember that when you come out of the Database you will probably be in Muscat, so will need to type 'stop' to return to the main computer operating system.

To enter the Database: the Database (DB) and Direct Access(DA) systems.

It may be that you decide to have separate DB and DA systems (as described above). If so, when you are in the CDS 2000 version of muscat (having gone in with muscat cds, as above):

for the DB system	type	c-dbsys
for the DA system	type	c-disc

To enter the database if you have gone into 'Discatel', type:

for the DB system	type	c-disc
for the DA system	type	c-dbsys

Turning the record numbers on and off.

It is possible to have record numbers at the top of each screen in CDS 2000. For instance, the first screen would then have 'Record A:1' at the top. It is often useful for editing and checking purposes to have the record number, but otherwise it is not necessary. So the default is to have the numbers turned off. If you want them turned on, type the entry command, followed by 'numbers on', for instance:

c-dbsys numbers on

If the default has been set with the numbers on, then type:

c-dbsys numbers off

The record numbers will then be printed on the screen (or omitted) as the case may be.

Monochrome and colour screens.

The CDS system comes in two forms, for colour and monochrome screens on computers. There are two aspects to this. The first concerns the mode of display on the screen. This tends to vary in attractiveness with the make of computer as much as whether it is a monochrome or colour screen. If you have a colour version, and want to set it to work in the monochrome mode, type:

c-dbsys with - or c-disc with -

as appropriate. This is a more satisfactory mode for a number of colour screens and you should try it out.

If the system comes up in the default of a monochrome version, then you will need to alter the 'macros' if you want a colour default. Go to the appropriate set of macros (\muscat\macros\cds or discatel or cdsi). You will find one or two files called disc.txt and dbsys.txt. With your word-processor, in non-document mode (i.e. with no control characters), take out the '-' (minus) sign which you will find in each of them after <with>. It can be put back if you want to turn the screen back to monochrome.

Secondly, the computer needs to know whether it is dealing with a colour or mono system. The default is assumed to be a colour screen, since most computers now have these. But if this is not the case, you will need to make a change. In the sub-directory containing the macros through which you will enter the system (\muscat\macros\cds or discatel or cdsi, depending on which you enter) you will find the dbsys.txt and/or disc.txt macros which were mentioned above. These need to be edited as follows with or without the letter z, as follows:

monochrome	**colour**
<with>	<with>
z	
!	!
disc file....	disc file....

By changing the dbsys.txt and disc.txt macros in this and other ways, you can alter the various defaults set when entering the

system.

Turning the videodisc on and off.

Depending on the default setting in c-dbsys, your system may assume that you are linked to and want to use a videodisc. If you find that you are assumed to want a 'genlocked' version, but want to turn the videodisc off for a while, then type:

```
c-dbsys mode 0
```

If you find that the videodisc picture is not shown on the screen, but would like it to be shown through the genlock, then type:

```
c-dbsys with v
```

One-screen and two-screen versions.

If you have a computer linked to a videodisc player by a 'genlocking' (Videologic 'Mic' card) device, then it is possible to see both text and picture on the same screen. It is quite an expensive option, however, because of the cost of linking devices.

It is possible to run the CDS in a two-screen version, using your normal computer screen for the text, linked to a videodisc through an RS232 cable, so that the picture is shown on a TV screen. To use this two-screen version most effectively, type:

```
c-dbsys opts 2
```

The two versions of the program to run the one-screen and two screen versions are held in the \muscat\mods directory as:

```
disctwo.cin             -   for two screen version
discmic.cin             -   for one-screen (genlock) version
```

Currently a copy of disctwo.cin has been made and called disc.cin. This is way drives the system. If you want to use the on-screen version, you should copy discmic.cin to disc.cin. (Do not use re-name, as it is necessary to keep back-up copy of the original, so that you can easily change back).

The number of records to be retrieved.

It is possible to limit or expand the number of records to be retrieved by specifying the number as you enter the system. The default is one thousand (1000) records, so that, for in-

stance, in a structured query you will sometimes get the message '1000 out of (say) 1775 records retrieved.' If you want to set the number of records to be retrieved at more or less than this, type, for example:

c-dbsys opts z2000

This will set the limit to 2000, rather than 1000 for the session.

Selecting which database to enter

If you use the ordinary c-dbsys or c-disc entry, you will go into your normal (default) database, which is DB.DA in the 'cds' sub-directory of Muscat. You can, however, have as many databases as you like, and they can be in any directory. It is possible to specify an alternative by going:

c-dbsys on (database name)

For instance, if you have a database called book.da, which is in a sub-directory called \work, you would type:

c-dbsys on \work\book

Note that the extension .da can be left off in this command. The computer assumes that a database will have this extension. It is thus essential that all database files have this extension.

It is, of course, possible to combine these various options. If you find that you are using a set of them very often and want to make them the defaults, you can alter the dbsys or disc macros.

Selecting which set of macros and which database to use

It is possible to be in any directory and to go into the appropriate set of macros and database in another directory by using the system.txt macro.

If, for instance, you were in a directory called 'letters' and you wanted to enter the database and set of macros kept in the \muscat\cds sub-directory, you would type the following:

muscat cds

c-system \muscat\cds

This would take you into the appropriate set of macros.

Likewise, using discatel, you could type, from any sub-directory:

muscat discatel

c-system \video

Then you would be able to use the macros in the sub-directory \video.

CHAPTER FOUR. SEARCHING THE DATA

CONTROLLING THE SYSTEM

The first choices on the introductory page.

On entering the database system, you will be presented with the following page:

Welcome to the Cambridge Database System

Press **H** for help

Introductory text:

Tutorials:

Contents:

Free text query:

Structured query:

Both free text and structured query:

The first three of these are fairly self-explanatory. Some preliminary advice on how to move the cursor around the screen and make selections is given by pressing H. Certain introductory information is given in the introduction. Tutorials can be built up on selected topics.

Controlling the screen and making selections.

Each screen you see contains several little boxes, or 'icons', one of which will be flashing or more highly illuminated than the others. You can change which box is 'active' (i.e. flashing or highlighted) by pressing the four arrow keys which are together on the keyboard. If you press the 'Enter' key (the large key which is on the right, where a carriage return key would be on a typewriter), you will select the active box. This is one way to make a choice to do something.

A box is often accompanied with a word beginning with a highlighted letter in a different colour or shade. Pressing the letter key selects the box. For instance, 'F' or 'f' will select '**F**ree text query' on the introductory page. This is the other way in which to make a selection or choice.

Moving back up the system.

Imagine that you start at the top of the system, and when you select a box, you usually go down a particular path or branch of a tree, each subsequent choice taking you down a further branch. To return back up to the previous level you press 'Esc' , or select the box with 'Esc' against it. Thus, if you are lost, go back to the start with Esc. In essence then, 'Esc' will return you to the screen from which you made the last choice, while the boxes and letters will take you downwards or sideways through the choices.

It important to be careful not to press too heavily or long on a key, as it may 'repeat' and you may be taken beyond the point which you hoped to reach.

To leave the Database system, return to the introductory page where 'EXit' appears on the bottom menu, and type the letter x.

If you are unsure of what particular choices on the screen mean, if there is a 'Help' key, this may be selected by pressing on the box or pressing the F1 function key' and you will be given a short description of each choice.

TYPES OF QUERY

Types of searching system.

There are four ways of finding materials with this system, by a hierarchical table of contents, by putting in a string of words through free text searching, by making choices from lists of words and combining these choices through 'structured' queries, and finally by combined free text and structured queries. Let us look at each of these.

A hierarchical table of contents.

Select 'Contents' from the introductory page, then your are first presented with a choice as follows:

By name of collector or author

By medium or source

If you select 'name', you might be presented with a list of the people whose materials are on either the videodisc or the

computer disc. Then, if you choose one of them, a list of their various works could be presented. If you select one of these, a list of the short captions which describe each item in that work is presented, and you can choose to see any of these.

Alternatively, you could select 'medium or source', which would then take you to a list organised by photographs, with the authors of each set, which could again be selected. This hierarchical contents system is the nearest equivalent one has to a table of contents in a book.

In fact, when you select an author or source, you set an 'embedded query' running, a query which you could set up for yourself once you know how the system works. The way these embedded queries work is described at the end of Chapter 2 above.

The indexing power of the computer, however, allows you to explore the materials in two other ways, which take you to the exact record you are looking for. This provides a set of combined indexes which is like a well indexed book having a set of indexes of place, person, subject, date, ethnic group, medium of recording. Then, having such a set of indexes, the computer makes it possible to combine them in any conceivable way, and finds the appropriate image or text more or less instantaneously.

Free text queries.

Free text queries allow you to put in a string of words which are matched against the records, and the best matching record is given first, the next best next and so on. What 'best' means will be explained later, but an example of such a free text query would be:

Free text query> green and blue hats

This can be made much more elaborate, or simpler.

These queries will search the 'caption' or short text part of a record, and the other keyword fields, except the longer text field. They are thus simple to make and serve for many purposes. If you select 'Free text query' in the initial page you can make such a query. Since the words in the index are 'suffix stripped' (explained in Appendix G below), it does not matter whether you put in 'dance', 'dancing' or 'dances'; all references to dance will be found.

Structured queries.

These are queries of the form 'and' and 'or'. You are basically invited to begin to build up a set of query terms, selecting from diverse lists of possible terms under various headings. For instance, you might choose a year, a person, an ethnic group and a medium as follows:

1939 Bower Zemi Photographs

such a query would find all the photographs of Zemi Nagas taken in 1939 by Ursula Graham Bower.

This is therefore a much more precise and powerful system. It delimits an area of interest. It operates on all the major fields other than the caption field.

Combining free text and structured queries.

Since the two types of query complement each other and act on different fields, they can be combined. In the example above, you could thus, by choosing 'Both free text with structured', enter the following:

Free text query> blue green hats

Structured query> 1939 Betts Zemi Photographs
This would locate any photographs of blue green hats taken in 1939 by Betts among the Zemi Nagas.

The base page.

The choice of free text, structured or both free text and structured queries leads to the appearance of a 'base' page, that is a screen to which you will frequently return. Each query will produce a slightly different base page initially. Furthermore, the base page increases in complexity as more features of the system are used.

If we take the 'free text' option as an example, it starts with a page which merely allows you to ask for a free text query or alter the retrieval style. When you ask a query, the base page then expands to allow you to inspect the free text query. When you have looked at some answers and 'marked' some of them as relevant, it allows the new option of keeping and/or expanding the query. If they are kept, you then have the option of seeing the marked items.

The meaning of all these options will be explained later.

The important point is that there is a 'base' page of increasing complexity which exists at the next level down from the 'introductory' page.

FREE TEXT QUERIES AND HOW TO MAKE THEM

How to make a free text query.

This is relatively simple. Having selected the free text query system and then selected Free Text Query, you are given a prompt at the top of the screen:

Free text query>

at which you type in one or more words. The distinction between upper and lower case is ignored, so proper names and places can be in either. You will then be shown a screen which has among its captions the option of 'Inspect the free text query' and 'Go to first item'. If you would like to see how many times the words you have asked for appear in the whole database, you can select the 'Inspect the free text query' choice. You would then get, for example:

blue 47
green 84
hat 275

Against each is a box allowing you to delete this term from the query. This may be necessary if you have chosen a term that is so common in the database that to include it in the query would give too many answers.

If you are satisfied with the query, it can then be run by selecting 'Go to first item'. After anything from a tenth of a second to half a minute, depending on the complexity of the query and power of the micro-computer, you will be shown the best matching record. You can then choose the next record by selecting 'new Item' at the bottom of the screen.

Order of being shown records.

The records which are found in free text queries are presented to the user in a probable order of usefulness as answers to the questions asked. For instance, if you have asked for four words, those records indexed by all four will come first, records indexed by only one will come last, and the rest will be in the middle.

The precise ordering of the records depends upon various information retrieval criteria such as the frequency of the

words occurrence in the whole database. Words which are used thousands of times are likely, all other things being equal, to be less useful than words which are only occasionally used; 'shaman' is probably more useful than 'man'. (For a fuller description of probabilistic retrieval, see Appendix. C).

STRUCTURED QUERIES AND HOW TO MAKE THEM

Setting up a structured query.

If you select the structured query method, you will be offered the following choices:

Year
Other date
Person involved
Locality name
Ethnic group
Medium of recording
Source of material
Videodisc frames

We may look at each of these in turn, starting with what would happen if you selected videodisc frames.

Finding the record relating to a videodisc frame number.

It often happens that you have seen a picture on the videodisc and want to find out what it is. By selecting 'Videodisc frames' you are invited to "Enter a frame number". If the enter key with no number is typed, you will be taken to a list of frame numbers, starting at the lowest. You can go up and down this and choose to see any of the numbers.

If, on the other hand, you wanted to find the record describing the image on videodisc frame number 625, type '00625' in answer to the prompt. As explained earlier, the computer assumes all numbers have five digits, so you need to make up the number to five digits by adding the appropriate number of leading zeros. For example, you might have 00007, 00015, 00625, 02751.

If you type in 00625 you would then be taken to a page of numbers. In this case, for instance, you might have a page which went:

00625
00626
00627-00629
00630 and onwards....

By selecting a number, you are taken to the record associated with that videodisc image. You can then go to the actual image

if you like. Frequently, especially with sets of still photographs or moving films, you will be given a range, as in 627-629 in the example above. Selecting this will take you to the start of a moving film or photograph set.

Finding dates; spans and years.

There are two principal methods of finding dates. The first is to set up a query by asking to see all records which fall within a specified year. This is done by selecting 'Year'. You will then be taken to a screen which asks you to select a certain year range, for instance:

Up to 1879
1880 - 1919
1920 - 1969
From 1970

Choice of one of these will take you to a screen which allows you to select specific years. As with all the structured fields, it is possible to build up a list of the 'or' variety. Thus you could select 1935 by itself, or you could select 1935,1936,1937. The effect of the latter would be a query of the form: find records with the dates 1935 or 1936 or 1937. (You select by pressing once on the appropriate box; pressing on the same box again cancels that selection. It is thus a 'toggle'.)

Such a query would now find every record which had the year date 1935 or 1936 or 1937 in it (whether more precise days or months were specified as well as the year or not).

Finding dates; days, months and years.

It may well be that you would like to look for more precise dates. This can be done by choosing 'Other date'. On selecting this, you will be requested as follows: enter as yyyy or yyyy/mm or yyyy/mm/dd>

This means that you can enter, for example, any of the following forms:

1934
1934/12
1934/12/20

In other words you can put in just 1934, or December 1934 or the 2Oth December 1934. When any of these have been entered, you will be taken to a list of dates. For instance, you might have asked for 1934/12/20 and be shown:

1934/12/20
1934/12/22
1934/12/27
1934/12/30
1935
1935/01

By selecting a date, you will set up a query to find all records of that date. Or you could select a series of dates, of the form '1934/12/20 or 1934/12/22 or 1934/12/27'.

Finding persons.

By choosing 'Person involved', you are invited to enter a surname. It is possible to type in a whole name, or just one or more letters of the alphabet. You are then taken to a list. If you had typed in, for instance, Bower, the following might appear:

Bower
Bower/ Ursula
Bower/ Ursula Graham
Bower/ Mrs Graham
Bowers
Brown
Brown/ Col

Again you can select either one name, or a list of names, such as 'Bower or Bower/ Ursula or Brown/ Col'.

Finding ethnic group.

On choosing 'ethnic group' you are asked for a name of an ethnic group or tribe. Having typed in a name, or even a single letter of the alphabet, you are again taken to a list. The procedure of choice is then exactly the same as person.

Locality or place.

On choosing 'locality', you are asked for a name, at which you can type in nothing, one or two letters, or a full name. If a full name is given, you will be taken to the list of place names, as in person and ethnic group, and can make a selection.

It may be that you will have data where synonyms need to be indicated. An example of how this works can be taken from one application of CDS, namely to the data about the Naga tribes of the north-eastern frontier of India.

Since Naga place names can be spelt in many different ways,

it is necessary to provide a kind of synonym list. Supposing that a place was usually called Aopao, but that there were a number of alternative names used, including Pongyo. If you typed in Pongyo, you would be taken to a list with the first item 'Pongyo = Aopao'. This tells you that the standard name for Pongyo is Aopao.

If you just want to see where it is on the map, and to find what other synonyms there are, select this. But if you want all references to the place which is sometimes called 'Pongyo', you would go out of the list of place names, and then select 'Aopao'. This would then find all places called 'Aopao', and all variants of that name, including 'Pongyo'. When you come to look at a map you will find the standard name (Aopao) on it.

This synonym list was set up automatically, using a set of computer programs, partly on a mainframe computer, partly on a micro. Details are available from the authors of this manual, but they are not described here since they are specific to a particular set of data and would need to be considerably modified for any other data.

Medium of recording.

On selecting medium of recording, you will be given a choice of the different media, which can be selected individually or as a list. An example would be as follows:

film	(colour and black and white)
photograph	(colour and black and white)
sketch	(sketches, paintings, drawings)
sound	(audio recordings)
artefact	(three dimensional objects)
map	(sketch map)

Source of material.

If you select this choice, you will be taken to a list of the names of archives, museums, libraries and private individuals. These are taken from the *acq *p field, in other words the acquirer of the materials. By selecting one and combining it with other queries you can, for instance, see all the spears in a certain museum.

Deleting or modifying the structured query.

You select a query term by pressing on the appropriate box. In order to delete that term if you change your mind, press on the same box a second time. In other words, the box is a 'toggle', that will select and de-select. This can only be done

while you are still on the page with the list of terms. If you want to delete a whole query or set of terms, return to the initial structured query choice page (with the boxes for Person, Locality etc.). Then type 'D' or the 'Delete' box at the bottom, and this will clear away the whole structured query.

'Inspect the Structured Query' allows you to see what your current query is, add to it, or delete it, as described above.

Running a structured query.

As you select different fields and add in terms, you will begin to build up on the right hand side of the screen a structured query of a certain form, for example:

1934 or 1935 or 1936 and photo or film
and Konyak Nagas or Zemi Nagas

The query can be run, by returning to the 'base' page and then selecting 'Go to first item'. As well as placing the first answer on the screen, you will receive a message at the top of the screen stating how many records which answer the query have been found, thus:

125 records retrieved

That means that another 124 are awaiting inspection. The convention is that only 1000 records will be retrieved. If more than 1000 answers were found, a message will appear to state that:

1000 out of 2540 records retrieved.

It is assumed that the normal user would grow weary long before completing a search of one thousand records.

It is possible to change the number of retrieved records, as explained at the end of chapter 3. Thus you could limit the retrieved number to less than one thousand, or increase it beyond that limit.

COMBINED FREE TEXT WITH STRUCTURED QUERIES

By selecting 'B' or the appropriate box on the introductory page, it is possible to combine free text and structured queries. For instance, you could use the structured query system as outlined above to select a year range and a medium. Then use the free text query to put in a few words of free text. You could, for instance, ask to see carvings of houses (free text) in photographs in 1934 (structured).

THE WAYS OF LOOKING AT THE ANSWERS TO QUERIES

The three modes of looking at the found records.

So far we have assumed that a user will want to go straight to the full record, which describes a visual image or a piece of text. This is 'record retrieval' style. It is possible to use other methods of looking through a string of records as an alternative to this.

When faced with a screen which gives you the option to 'Alter the retrieval style' you can select that choice. You are then given two further options: to select 'Data retrieval' or 'Caption retrieval'.

Data retrieval style.

If you select 'Data retrieval', instead of being taken to the record describing an item of data (a picture or a text record), you will be taken to the data itself.

If you want to see the record by which you have entered the image or page of text, select 'r' from the menu. If you want to return in data retrieval mode after looking at the record return to the data level by typing 's' for show.

You can move through the data in two ways. If there is a 'Prev' or 'Next' box on the menu bar, then you can go to another item of data (e.g. another photograph) directly by selecting 'p' or 'n'. This would be the case, for instance, if there was a set of photographs referred to in a record.

Or you can go to a new Item of data by selecting 'i'. This might then take you to some more film, a page of text, another still image, which might in turn contains further Next and Previous choices.

The items can be 'marked' with the appropriate box and the menu bars turned off and on with the 't' or text on/off choice.

Quite often you will be taken to what is obviously a record describing a set of images or pieces of moving film, each of which has a separate caption. Here you choose the one you want to see by selecting a box, which takes you to the data, consisting of a still picture or moving film.

If you want to see the record by which an image or moving film or piece of text is indexed, type 'r' for return. You can go back to the image with 's' for show. If you find yourself at

the level of the records, and want to go on looking at images, go 's' to go back into that level.

Caption retrieval style.

If you select 'Caption retrieval' you are in 'caption retrieval style'. If you have set up a query, and now select 'Caption list' from the menu, instead of being presented with the first record, you will be given a screen of short captions, for instance:

A bow and arrow for shooting at fish during a fishing trip.
A young man out fishing, using a bow and arrow.
A group of men and women out on a fish poisoning trip.
A fish trap used for catching fish after poisoning.

Each short caption has with it a box which allows you to select it; in which case you will be taken again to the full record from which the short caption was taken. In this way it is possible to scroll quickly through pages, referring to dozens of records. The short captions are always automatically truncated to 69 characters, i.e. one line on the screen. If you 'mark' a record (as explained later), an asterisk appears against the marked record in the caption list.

In order to return to the 'Retrieval by record' style, you again select 'Alter retrieval style', and select 'Retrieval by record'.

MOVING BETWEEN RECORDS, IMAGES AND TEXTS
Moving from record to record.

Let us assume that you have asked a query and gone to the first answer. You will then have a menu bar at the bottom. Among the items on this will be 'new Item'. This will take you to the next record answering the query. If you go to that, a new possibility appears on the menu bar, Previous. This takes you back to the previous record, already seen.

If you go back to that, you now have a further possibility, 'Next'. That will take you on to the next record, i.e. a record that has already been seen. If you want to skip to a new, as yet unseen, record, you select 'new Item'. Once you have a list of records that have been seen, a further possibility opens up, namely 'First'. This will take you back to the first record you saw while doing the present query.

This can be illustrated thus:

```
record 1          selecting 'new Item' takes you to...
record 2          selecting 'previous' takes you to...
record 1          selecting 'next' takes you to...
record 2

or
record 1          selecting 'new Item' takes you to...
record 3          selecting 'first' takes you to...
record 1
```

Seeing a picture or piece of text.

Let us assume that we are now at the level of a specific record. This may be complete in itself, or it may refer to an image on the videodisc or to a piece of text. If this is the case, there are two ways of moving to the image or text.

If there is only one image or text referred to, a 'Show' request will appear on the menu bar. By selecting this, you will be shown the relevant image or page of text. To return from the image or text, you type r, for Return.

If there are several cross-references to images or texts, you reach them in a different way. In this case, several highlighted boxes (icons) will appear on the screen and by selecting one, that particular image or text will be shown. This is the case, for instance, where there is a general description of an event, of which there is a set of discrete images, each with its sub-description.

For instance, there might be a record as follows:

```
The Spring festival dance at Longkhai village in April 1937
     - getting ready for the dance
     - dressing of the boys
     - dressing of the girls
     - dancing      - the drummer playing
     - further dancing
```

You would have a series of choices, and could see each section as you liked, going straight to the dance, or the drummer.

It should be noted that occasionally there will be a list in which there are some lines where there is a box with no text beside it. This means that the general description at the top of the record is an adequate description for that entry: thus the record above might have looked like:

```
The Spring festival dance at Longkhai village in April 1937
```

* getting ready for the dance
* dressing of the boys
* dressing of the girls
*
* the drummer playing
*

Examining moving film.

If you are taken to a piece of moving film by any of the retrieval strategies, you will first see a still frame with 'Film: (first frame on show)' at the top and at the bottom a menu bar with Help, Return, Show and Mark.

If you want to see the film, press 's' or the 'Show' box and the film will be shown through at normal speed, with a message at the top saying: 'Film starting at 2O55O and playing through...' (or whatever the appropriate number is). On the menu bar at the bottom will now appear Return, Stop, Go from start, Text off.

By pressing 't' for text off/on, you can turn the overlay text off and on. If you reach the end and want to see the film again, press 'g' or the 'Go from start' box. If you press 's' or the Stop box, the film will be stopped or stilled at that frame and you will be given the frame number at the top.

On the menu bar at the bottom appear the following choices: Return, Go, Backwards, Speed 1, Prev frame, Next frame, Mark. You can mark the still frame in the usual way. You can move forwards (step through the film) or backwards with 'n' and 'p'. If you want to change the speed, type '1' and you will be in slow motion (typing 2 will restore you to normal speed). If you want to see the film backwards, press 'b' and then 'g'. Select 'f' and then 'g' to set the film moving forwards again.

Reading texts.

If you are taken to a text record when you press 'Show', for instance a paragraph of a diary or book, it is possible to read through the previous and next paragraphs, even to read the whole work from beginning to end, by using the 'previous' and 'next' boxes (or typing 'n' or 'p'), at the bottom of the screen.

If you do this, it is possible that you will find flashing boxes embedded in the text. These also appear in records. These are cross-references to images on the videodisc. By selecting them, you will be taken elsewhere, to a photograph or sketch made on that day. Having seen the cross-reference, an 'r' or

return will bring you back to the paragraph you left.

Listening to sound.

A videodisc can also contain sound; speech, music and other material. In order to find this sound, you would need to type in a query with such words as song, music, sound, interview. If you come to a record which clearly describes sound, you select the 'show' box on the menu bar. This then presents you with a blank screen with another menu bar with a 'show' box on the screen. Pressing this starts the sound. If you want to interrupt the sound in the middle, type 'r' for return while it is playing.

EXPANDING QUERIES; THE MARKING SYSTEM

We have seen that answers to your questions will be arranged in what the computer thinks is probably the best way to answer the query. This gives a first approximation, but there is also an option to enter into an interactive dialogue with the computer to improve the query. This is known technically as 'relevance feedback with automatic query expansion'. It is this, along with the combined free text and structured system, which makes the database unusually powerful.

Seeing the terms by which a record is indexed.

When you have been shown a record, one of the possible features to select on the menu is 'Terms'. If you select this, a complete list of the terms by which that document has been indexed will be shown.

It is possible at this stage to add terms from this list to the query, which can then be re-run with an 'expanded query'. For instance, you may be looking for 'blue green hats' as the free text query. You may then be shown the first record found. This may show a list of indexing terms that includes bamboo, hat, tassels and so on. You might think that it would be interesting to add in 'bamboo', and re-run the query. This can be done. This is a simple form of query expansion which acts on a particular record.

A more powerful version of this, acting on a set of selected records, is called 'full relevance feedback and query expansion'.

Full relevance feedback; marking the records.

This only works in the 'free text' retrieval system. When

presented with a record, one choice on the menu is the word 'Mark'. If you select this, it is noted that the record has been 'marked'. Basically, you have said that this record is indeed the sort of thing you were looking for, it is indeed a 'relevant' answer to your query. (If you change your mind, you can select what is now an 'unMark' choice, in which case it will no longer be marked.)

You then move to the next record and can again mark the record if it is a good answer to the question. Thus you go through, putting into a special file a set of 'marked' records.

When you return from looking at particular records to the 'base' page, two further choices will be offered. One is to 'Keep the marked items'. If this is selected, a request for a name for the file of marked items is made. You might have marked a set of photographs about fishing, for instance, and title this 'fishing'.

Having given it a title, you will be given a new option, namely 'See the marked items'. If this is selected, a short caption list of the marked items will be presented. In each case, an opportunity is presented either to see the marked item again, or to delete it from the marked items file.

If you go into 'caption retrieval' mode, or have been working from this mode, you will find that as records are marked they are given an asterisk on the list.

Supposing there is now a marked file containing six answers to a query about fishing. You may want to improve the query. This is done by selecting 'Expand the query'. The computer will then take all the records marked as relevant answers to a query and examine them. It will print out a list of the terms by which they were indexed.

This will be presented to you in an order of probable usefulness. In other words, terms that are very highly associated with the marked records and occur infrequently in general in the database, will be presented first. You can add in any of these terms to the original list and re-run the query. This is query expansion, following relevance feedback.

An example would be as follows. Supposing you had marked four records as relevant. You might find a list of terms as follows:

```
red     freq 47
river   freq 96
konyak  freq 250
stone   freq 35
```

```
bamboo  freq 276
man     freq 876
blue    freq 36
house   freq 125
woman   freq 176
bridge  freq 24
jump    freq 36
head    freq 230
```

This is an over-simplified example; the list is likely to be longer and more elaborate. But what it shows is a pattern. The words 'red', 'river' and 'konyak' have been found in all four of the relevant, marked, records. They are thus placed first, with the one that occurs least frequently, namely 'red', which only occurs 47 times in the whole database, first, then river, then konyak. Then we have the terms which occur in three of the records, namely stone, bamboo and man, again placed in order, with the least common first.

What the computer is doing is showing statistical connections to other words. It makes it possible to see, in large data sets, that there are connections of a kind you might not have noticed by merely looking at the separate records. If you decide that the suggested connection is interesting, it can be added in. Thus you could add in 'red', 'river' and 'konyak' and having expanded the query, re-run it. You are now likely to get a somewhat different set of records which may provide new, and possibly more fruitful answers. These again can be marked and expanded.

One word of caution. If you are going through a set of records and marking a large number of records, you may suddenly receive a warning that you are running out of space. This happens because of the following. If you mark a record (but not a picture), then the computer stores all the terms by which that record is indexed in core. If you mark a large number of records, say a hundred or so, they may generate many terms. This may use up all the available core.

If this becomes a serious problem and you want to allocate more work-space, this can be done by following the instructions about workspace allocation in Appendix F, under 'Calling Muscat'.

This is thus an interactive process of searching, using the reasoning and associative knowledge of the human, and the speed and statistical power of the computer, alongside each other.

For many kinds of research, whether in anthropology, medicine, sociology, history or elsewhere, this is a powerful

research technique, based on very simple premises.

SETTING UP FILES OF MARKED ITEMS AS TUTORIALS.

We noted that 'query expansion' could only be done with the free text query part of the interrogation system. On the other hand, there is another use for the 'marking' system which explains why one can 'mark' records in structured queries, and even mark bits of moving film or photographs, or sound.

It is possible, using the marking system to set up a file of records which may be used to create pathways through the materials. These may form the basis for files which, through editing, are used as future tutorials. We may explain briefly how this can be done.

Two forms of marking and captioning.

We have seen above how to mark records and then to keep them and give the file a title. You may also want to mark the images or even texts themselves. In this case, having found the image, for instance a photograph, you select 'mark'. On this occasion you are asked to provide a title. You then type in a short description of the image, which is stored in the marked file, rather than the record itself. Thus you might have a list of marked photographs with captions:

a good photograph of a man cutting wood
some men dragging a huge pole through a forest
an axe blade
men throwing logs up onto a pile of wood

These can then be saved, given a file name, and examined (seen or deleted), as with a normal marked file. It is possible not only to mark sections of moving film, but also to mark single frames of such film, selected from a moving sequence.

Saving, editing and re-entering the marked file.

When you have marked a set of documents and kept them with a name for the file, you will see a choice called 'Inspect marked file'. If you select this, you will be shown a list of the marked items. On the menu bar at the bottom two new choices, Input and Output will appear. If you want to save your set of marked records to a permanent file outside the database, select 'output'. You will be asked for a name. Choose a name, and press carriage return. The set of marked records will then be written to a permanent file, with the extension .mks, in whatever directory you entered the system from.

When a marked file is 'output' in this way, it is deleted from the database. But if you want to retrieve it, or retrieve other marked files, you can select 'input' at the bottom of the screen, or the 'Input marked file' choice on the base page screen. If you select one of these, you will be asked for the name of the file to be input. When you specify a name, that file of marked items will be added in.

Thus you can output and input sets of marked items from a library of files that you build up on your hard disc.

You may find it difficult to remember the names of all your marked files. One easy way to see what is in your library, is as follows. While in the database, go into muscat with the Alt key and F8 key combined. You can now do any of the MSDOS or DOS commands, by prefacing them as appropriate with msdos or dos. Thus, if you wanted to see what was in your current directory, you could type, for example:

msdos dir /w (which will give you a wide listing)

When a file has been put out of the database in this way, you can turn it into an ordinary muscat file:

In the case of a db.da file type:

c-getmrecs filename to filename

Or, if you have compressed your material into a direct access (DA) system, then type:
c-getdiscm filename to filename

It is then possible to turn it into a text file by typing:

c-print filename to filename (or c-list)

You could then edit it (within the same set of conventions and print commands), and build it again and give it the extension of .mks and add it back into a database. This is a way of building up your own tutorials.

PART B: APPENDICES

APPENDIX A.

Martin Porter

Technical details concerning cross-references.

The cross-reference may take one of three forms:

\A.n\ (n . 0, structure query only)

\A.0\ "s" (s is some text, free text query only)

\A.n\ "s" (structured and free text query)

You can then select one of these, which will take you to the standard page for query entry. If \A.n\ is supplied with n > 0, structured queries will be enterable, with record A.n giving the base record for controlling structured query entry. If "s" is supplied, free text queries will be enterable, and analysed according to the contents of string "s".

Contents of the string "s".

The string "s" can contain:

s - stem with a stemming algorithm

v 's' - treat all characters except alphas and those in 's' as gaps

i 's' - treat characters in string 's' as ignorable

j 's' - treat characters in string 's' as ignorable escapes

g 's' - treat characters in string 's' as gap characters

p 's' - use 's' as a prefix for extracted terms

l - force lower case

u - force upper case

c - force initial capital

This follows the usages in the i-directives in INDEX. Note that a v-option must precede any i-,j-, and g-options; that the l- and u-options should not be used together, that a c-option forces initial capital after an l-option has put the whole term

into lower case, and that stemming only takes place on lower case words, so that it is usual to use the s- and l-options together. By default, space is a gap character, while all other characters go to form terms, and the prefix string is null.

APPENDIX B

Some suggestions on data entry and error corrections.

If you are engaged on a large data entry exercise, you may wish to consider various ways in which to speed up the entry of text.

There are two aspects to this. There is firstly the entry of the text or data itself. Here the main choice is whether to put the material in by typing it, or have it optically scanned by using some form of optical character recognition by computer (OCR). The cost of the latter method is dropping rapidly and the efficiency increasing. A number of devices can now be attached to ordinary micro-computers in order to achieve this.

Once the data is in, or while it is going in, the field codes ('tags' or 'flags') need to be added. How this is to be done will depend on many factors, but four methods can be mentioned.

In all these methods, remember that it is essential that the final text of the material to be put into the database must not contain the control characters which are added by a number of word processors. To avoid these characters being included, depending on your word processor, select the 'non-document' mode when you start to edit, or the 'non-formatted' mode when you end.

Simple input method.

You can just type in the appropriate codes as needed, for instance *kp, *kd *z etc. This is useful if you are editing a file which has already been created, for instance a long set of texts which have been generated by optical character recognition input. Obviously, if you are using this method, the more 'mnemonic' or memorable the codes are, the easier; for instance *p for person *d for date. Likewise, the shorter the codes are, the quicker it will be.

Setting the function keys.

If you are using the method above, you may find it quicker to set up the function keys on your micro to correspond to the most frequently used codes - thus F1 might correspond to *kd etc. Setting up function keys will be explained in the manuals for your word processor.

Setting up a menu of codes.

If a set of records in a file has a fairly standard set of fields with associated codes, the quickest way (and also a useful reminder to include material under all the fields), may be to set up a small file on your computer which provides a 'template' of the codes. This can then be inserted simply each time you start a new record by using the 'insert block' or 'insert file' command on your word-processor. Thus you would be given a series of codes starting each line, and then type the data alongside these. CDS does not object to empty fields, so it does not matter if not all the fields are filled with data.

Setting up a screen with 'boxes' for data.

It is possible to take the previous method one stage further by setting up more elaborate 'boxes' for data input. These have the field type specified, and allow you to type in a restricted or unlimited amount of material into each 'box'. The 'card image' on the screen would then have as its output a file with the appropriate codes added in to the record. A way to do this is available in the CDSi software available with this package, described in the accompanying CDSi manual.

A method of coding longer texts.

As larger quantities of texts become available through optical character recognition input and from previous machine-readable files, a method of preparing this for input to the database with minimum coding is helpful. One such method is described in Appendix H, namely a utility for allowing one to add codes within a longer text and then run a program to automatically extract the coded items, as appropriate. This saves a great deal of double typing.

Some guidance on how to correct errors thrown up by 'build'.

In a number of checking procedures, and particularly in

'c-build' you are bound to find errors. The error messages take a form such as the following:

Line 155 Character t (at position 8) found while reading integer
Line 548 Character p (at position 16) found while reading integer
Line 559 Character k (at position 10) found while reading integer

This tries to specify as clearly as possible the error, but, of course, the error may be caused by an earlier mistake. In particular, hundreds or thousands of errors may be caused by one faulty 'global' setting using the 'setting constants' method.

In order to look at the precise error, you need to find the offending line. If you have a line editor, you can use that, since it will have a 'move to line n' command. Or you can use a word-processor. Each word-processor is different and the documentation about line numbers is often weak. You can often not issue a command to find a line number. But the line-numbers are indicated.

To give one example, from the word-processing system 'Word-star'(c). In this word-processor, if you enter in what is known as 'document' mode, you will find that the line-numbers start afresh each page. If you wanted to find line number 247 in this mode, it would be best to set the page length, say to 100, then move to the 47th line of the third page. A simpler method is to enter in 'non-document' mode. The pages are not split up, and with a certain amount of 'help' on, you will see the lines at the top of the page. You can rapidly page through to the right line number and make the correction.

APPENDIX C

The basis of the probabilistic retrieval system.

The 'probabilistic' retrieval system is only associated with the 'free text' query system. 'Structured queries' are equivalent to the relational database models upon which most databases work, namely exact matches. Since probabilistic retrieval is an important feature of this system, it is worth looking briefly at the probability theory model of information retrieval.

In essence one puts in a list of index terms and asks for

close matches to this list. The answers with the best match are shown first, followed by those with a less close match and so on. But what does one mean by a 'best match'?

Each term in a query is first given a weighting, described as "a kind of measure of importance" (MI,p.138); the weights are recomputed each time a new query is set up. In other words, the weightings are not fixed, but dynamic, being a combination of the frequency of the word in the database (hence the random statistical probability of the term appearing in any given answer), plus the degree to which the term is associated with the 'relevant' documents. In other words, a term like 'man' would tend to have a lower weighting (it is less valuable as an indexing term because it appears very often), than a word like 'opium', which occurs infrequently. All else being equal, the higher the frequency, the lower the weight. That is one element.

The second element in the weighting is the degree of association to records marked as relevant. In other words, if five records are marked as 'relevant', and 'opium' only appears in one of them, but 'man' is in all five, notwithstanding its common occurrence, 'man' will be given a higher weighting because of its high association with relevant answers.

When deciding which document to show to the viewer first, the weighting of each word is added up, and those with the highest score are shown first. For instance, supposing one had two documents as follows (where the figures in brackets are hypothetical weightings):

record A opium (20) man (40) house (45) = 105

record B opium (20) marriage (10) man (40) = 70

Obviously record A would be shown to the user before record B. Likewise 'house' would appear before 'man' in a list of terms.

The addition is only done on index terms in records which are among the query terms. This is obviously necessary, since otherwise the records with many index terms would always come first.

The algorithm for working all this out, which is done before each new query is run, is given in MI,pp.159-160, and is as follows.

" Where 't' is the indexing term:

n = the frequency of term n
N = the total number of records in the database
r = the number of records in set R indexed by t

R = the total number of records in the set R

Then the weight for term t is given in the formula:

$$\log \frac{(r + 0.5)\ (N-n-R+r+0.5)}{(R-r+0.5)\ (n-r+0.5)}$$

(This is a logarithm to base e). The first match is obeyed, R and r are zero, and the formula simplifies accordingly. In presenting terms to the user in the 'terms' command, the value r/R-n/N is computed for each term, and the terms are arranged in decreasing order of their value. "

APPENDIX D.

How to make sequential searches outside a database.

There are two major programs for sequential searching outside the database. These are called RETX and KEYX and are described in detail in MM, pp.114ff. Here we will just give a few examples of how you might use RETX (sequential retrieval).

You start by going into MUSCAT and the first part of the query is set up by typing:

```
RETX   datafile to answer exp
```

where 'datafile' is a file of built records, and any records that match the query will be in the file called 'answer'. 'Exp' is short for expression, and the computer then moves to the next line and waits for some instructions after the expression.

You first ask the computer to search by typing 'find', then you define the field or fields in which it should look, and then specify the string of letters to be found.

In a simple case you might be looking through a list of photograph descriptions to find all those containing the word 'hat'.

You would type:

```
retx data to answer exp
find *u s eq 'hat' !
```

meaning "find the field *u string equals 'hat'" . The exclamation mark, !, tells the computer that the search expression is ended. It will then say how many records have been

found matching this query. These answers are contained in the file 'answer'. To read them, you have to type:

```
print answer to answer1
```

Answer1 is a new file, in text form, which can either be printed out on the screen or on a line printer.

There are several obvious ways in which you might want to build up more powerful queries. This can be done by building up queries of a structured kind, using 'and', 'or' and 'not'. Using 'or', for instance, you might put together a list as follows:

```
retx data to answer exp
find *s eq 'hat' or eq 'house' or eq 'gun'
!
```

Any record which had one or more of these words would thus be saved in 'answer'.

If you wanted to find a record which had several words concurrently, you could use the 'and' command, thus:

```
retx data to answer exp
find *u s eq 'picking' and eq 'millet'
!
```

You would then only get records which contained both words.

The expressions can be as long as is needed. You might also want to search for a combination of information in different fields. For instance, you might want to find the records of photographs taken by a particular photographer in a particular village. This could be done as follows:

```
retx data to answer exp
(find *kl s eq 'Ungma village') and (find *prod *p s eq
'Hutton')
!
```

(Note that there is no carriage return after 's eq' above; the find expression can be of any length, without carriage returns.) You would thus have all the photographs which mention Ungma village and the person Hutton.

To this could be added the third feature, 'not', which is added in as above, and will mean that you can exclude certain records, defined by another string.

Obviously such sequential searching is less easy than

searching within a structured database. It may, however, be useful in the early stages when you are editing and cleaning the materials before a database has been set up. It only needs individual files of built records, not a fully constructed database.

APPENDIX E

A parallel system working in 'q'.

The CDS 2000 interface allows you to set up a user-friendly way of undertaking and displaying information retrieval. It accesses a structured database which can also be accessed using the 'q' or query system (as described in MM, pp.150ff).

Thus, for instance, if you wanted to do so, it is possible to enter the list of all the terms in the index to the whole database by going into q from the muscat> prompt:

c-q

If you then type

i followed by a word, you will be taken to the nearest term to your query in the database.

i sun for instance might print

8: sun

This would first show the frequency of the occurrence of the word in the whole database (8 times here), then the word. To see the next word, you would type a carriage return, which might bring up the following (if repeated a few times):

8: sun
1: sunburn
2: sundai
11: sung

and so on. Note that the words may sometimes look rather odd, as they have been 'stemmed'. Thus 'sunday' appears as 'sundai'. The stemming system is explained in appendix G.

To finish looking at such a list, type 'q' for quit. If you want to move backwards through the index, type 'r'. If you want to find the records which are indexed by that term, type 'p' for postings.

If you just type an i, followed by a blank, you start at the top of the list, with the first index term in the database. These terms are the ones taken from the short title, or *u field, and from the *k or keyword field. If you want to find words in the specialised keyword fields, you can do this by prefixing the word after 'i' with an appropriate capital letter or number.

In our system these are:

E= ethnic group or tribe
L= location or place
P= person D= date
M= medium
S= source of material
V= videodisc number

Again if you just type, for instance,

i P= by itself, you will be taken to the list of persons

but if you type, for instance,

i P=Hutton you will be taken to 'Hutton' or the closest matching name in a list

In the case of V or videodisc number, you would obviously need to type a number. For instance you might type:

i V=00014 or i V=50073 or i V=50074-50079

It may be the case that the number you chose is in the middle of a sequence, in which case you will be taken to the nearest number that can be found.

With dates, you need to type in an exact date, of the form:

i D=1872 or i D=1872/12 or i D=1872/12/27

Which will take you into the appropriate part of the date term index.

A new feature of the 'q' system is that by going into 'q' and then typing 'info', you will be given details of how big the database you are in really is, in particular the percentage of it that is filled. Remember that while a DA database is by definition full, a DB database needs some spare space and is likely to start to refuse files at any point between being 85 and 90 per cent full. It will not fall over, but will just refuse to add the records.

APPENDIX F

Muscat on MS-DOS.

Martin Porter

File names in Muscat.

Within Muscat, the following extensions are associated with the following types of Muscat file:

type of file	extension
text file	TXT
Muscat sequential file	MUS
DA or DB file	DA
PAG output	PAG

In a Muscat command, the appropriate extension is added on to a file name if no extension is supplied by the user. Thus

```
build a:r\data to recs with spec.dat
```

is equivalent to

```
build a:r\data.txt to recs.mus with spec.dat
```

To refer to a file which has no extension, the file name should simply be terminated with '.', which tells Muscat that the extension is null. e.g.

```
build sample. to recs.
```

This builds from the file 'sample', with no extension, to the file 'recs', again with no extension.

The Muscat system is resident in a single directory, which we will denote by M. The name for M may be chosen by the installer, but by default it will be C:\muscat. Any files in this directory may be referred to in a Muscat command by putting '$' at the front of the name. Thus

```
$macros\el\build.txt
```

refers to the file el\build within directory M (by default, the file C:\muscat\macros\el\build.txt). This will not generally be

of interest to the user, except to note that $work gives the name of a directory which can be used for scratch files. By convention:

```
    $work\fa, $work\fb ... are scratch text files
    $work\fr, $work\fs ... are scratch Muscat files.
```

Muscat DA and DB files have a default block size of 2048 bytes.

Command set u.

On entry to Muscat, a command set with set letter 'u' is set up. 'u-help' gives a list of the commands available:

```
u-select application/a/r
    - selects the given Muscat application (e.g. 'u-select el')
u-dump from/a/r
    - does an indented-style print of the 'from' file on screen
u-sort from/a/r,to/k/a/r,opt
    - sorts 'from' to 'to' using 'opt' as the 'with' file. 'to2' etc
      are set up as temporary files
u-merge from/a/r,from2/a/r,to/k/a/r,opt
    - similarly, does a merge of 'from' and 'from2' to 'to'
```

The important one is 'u-select'. This is followed by an application name which selects a given Muscat application. If the application name is 'el', Muscatel is selected. The same effect can be achieved by entering Muscat with the command

```
muscat sys el
```

Muscat applications are organised so that they can be called up in this way. To create your own application called S, create a directory S in the muscat directory M\macros, and put your macros (with extension TXT) into S. Then put a macro file INIT.TXT into S. This will be obeyed when 'u-select S' is typed, and should be designed to set up an appropriate set letter for the rest of the command set, and possibly select a format. e.g.

```
||
alias c $macros\S\
format $formats\S
```

Equally 'muscat sys S' causes this file to be read as an INIT file.

(Here '||' is treated as a comment when the file is read as an INIT file, and as insertion brackets surrounding a null command

syntax when read as a macro. See below.) You do not have to use this convention, but it provides an easy way of setting up and distributing new applications.

Muscat comments in commands.

It is stated in 1.12.4 of the Muscat Manual that comments begin with the character '\' in the command language as well as elsewhere. In MS-DOS this proves to be unsatisfactory, since '\' is a significant component in file names. In the Muscat command language therefore (BUT ONLY IN THE COMMAND LANGUAGE) '|' replaces '\' as the comment character. e.g.

```
| Now retrieve recs with *on-fields
retx \dir1\in to \dir1\out  | here stream follows
goto *on  \ search for *on-field
!
```

This is the only incompatibility between the Muscat Manual and the MS-DOS implementation.

Calling Muscat.

The MS-DOS command to invoke Muscat has the form

```
muscat [[sys] S][init F][from F][log F][dir M][workspace N]
```

in other words, 'muscat' may be followed on the same line by a list of zero or more keyword-argument pairs, the keywords being 'sys', 'init' etc, and the arguments being single words, represented here by S, F, M and N. Here are some examples:

```
muscat sys library
   muscat init \d\init from \d\prog1            muscat
workspace 3000
```

S stands for a Muscat application (see 3), F for file names (see 1), M for the directory containing the whole Muscat system (see 1), and N for a number.

The FROM keyword specifies the source file of the Muscat commands to be obeyed. If omitted, Muscat commands are read from the terminal. The INIT keyword specifies a file of commands which will be obeyed before execution of the main command file given by the FROM keyword.

The SYS keyword is like INIT, but this time the source file of commands is taken to be macros\S\init.txt. Note that 'SYS' may

be omitted, so that

```
muscat el
```

is equivalent to:

```
muscat sys el
```

The LOG keyword specifies the file to which the Muscat log output is to be sent. If LOG is omitted, output is directed to the terminal. In the case of the log file, Muscat does not attempt to add on the extension TXT to the supplied file name.

The DIR keyword identifies the directory M. By default M is C:\muscat.

The WORKSPACE (or WS) keyword specifies the amount of RAM, in words (one word = 4 bytes), which is made available to Muscat. By default five eighths of the largest contiguous free area of RAM is used.

Command MSDOS.

The Muscat MSDOS command causes the text following the word MSDOS to be obeyed as an MS-DOS command, e.g.

```
Muscat> msdos xcopy c:main\s a: /p/e/s
```

Breaking.

To cause an attention in Muscat press CTRL-BREAK (i.e. hold down the CTRL key and press the key marked BREAK). Occasionally it may be necessary to follow this with the RETURN key. The response from Muscat will be:

```
quit, halt or ignore? (Answer Q, H or I)
```

Response I then continues the interrupted process. Response H halts the current command, and the user sees the 'Muscat>' prompt. Response Q quits Muscat.

TIME command.

A TIME command is supplied, but it prints out the time of day rather than the CPU usage. Even so, it can be used for its primary purpose, namely to get time estimates of Muscat operations.

Muscat extensions (to the Manual 3rd edition).

(i) The b-option in build can now have as a non-group code as the
code for analysis. Thus:

```
b *X{0} C *X{1} *X{2} ...
```

where C is a character and the *X{i} are non-group codes, enables

```
*X{0} D{0} C D{1} C ...
```

where the D{i} are text sequences, to be used as an abbreviation for

```
*X{0} D{0} *X{1} D{1} ...
```

(ii) DB files can now operate with k1n and k2n identity profiles (see section 5.2 of the manual).

(iii) The command REENTER causes Muscat to be re-entered with a new command level. A subsequent STOP returns the user to the earlier level of Muscat, with the aliases and format at that level restored. REENTER may be followed by a string which supplies the prompt, e.g.

```
Muscat> reenter p>
p> ...
p> stop
Muscat> ...
```

REENTER may be usefully called from a macro command, or with 'muscat reenter' from the Q command (see (iv)).

(iv) 'Muscat' command in Q. In the command Q, a command of the form 'muscat C' causes the Muscat command C to be obeyed. On completion return is made to Q, with the environment of Q undisturbed. e.g.

```
Q> rto recs.out
Q> Muscat c-list recs.out to recs.printed
Q> ...
```

(v) Muscat now provides store files. These are files which sit in the Muscat freespace, and vanish on exit from Muscat. Any file name beginning '!' is treated as a store file. e.g.

```
Muscat> copy to !spec from
....
```

```
    !
    Muscat> print recs to !out with !spec
```

The command DELSF (with up to 11 arguments) can be used to delete one or more store files, e.g.

```
        delsf !a !out-rec !temporaryfile
```

The command EXSF examines the currently existing store files, reporting names, types and sizes (if above 1K). (vi) The program NUMBER substitutes numbers for 'macros' in a text. It is particularly useful for numbering records of BUILD input. There are 26 variables a to z. A variable may be set by

```
    #(X=expression)
```

and substitutions made by including 'macros'

```
    #(expression).
```

Here X is a letter, and an expression is a list of letters and numbers separated by + or - signs. e.g.

```
    #(x=x+1) #(y=y-1+x) #(x) #(0-y)
```

A macro #(X') will be interpreted as #(X)#(X=X+1).

A typical call is:

```
        number in.file to out.file
```

The final value of any used variable is reported on completion of the command.

Transporting Muscat applications.

A Muscat application (e.g. Muscatel) is normally implemented as a set of macro commands residing in a single directory. It is handy to be able to pack the macros into a single file, either for editing purposes, or when one wants to implement the application on a version of Muscat running on a different kind of computer. As an aid to doing this, two Muscat commands, called PACK and UNPACK have been provided for transferring a group of files into and out of IEBUPDTE format (a format which we will explain below, but which will be familiar to users of IBM mainframe systems).

PACK has the command syntax filespec/a,to/k

The 'filespec' argument is an MSDOS 'ambiguous' file spec, e.g.

'C:\musc\macros\lib1*.txt' or 'mac\list?.txt',but the extension part must be specified explicitly without wildcard characters. The 'to' argument is an output text file which will contain all the files corresponding to the file spec, each file prefixed by a line of the form:

```
./ ADD NAME=x
```

where x is the name of the file (not including the extension). The 'to' file ends with the line:

```
./ ENDUP
```

The subfiles in the 'to' file are sorted into name order.

PACK leaves a sorted list of file names in the store file ! file_list. If 'to' is omitted, this is the only action of PACK, which is then like a variant of the MS-DOS DIR command.

UNPACK has the command syntax from/a/h,to/k/a,suffix/k

UNPACK is the converse of PACK. 'from' gives the file in IEBUPDTE format, 'to' and 'suffix' supply two strings, which, when placed before and after each member name x, define the destination file for that particular member. Thus the following commands are converses:

```
pack   C:\musc\macros\lib1\*.txt to flat1
unpack flat1 to C:\musc\macros\lib1\ suffix .txt
```

Note the character '\' at the end of the argument for 'to' in the unpack command.

APPENDIX G

Some features of the indexing system.

Suffix stripping, or stemming.

Reference has been made several times to 'suffix stripping'. This should be explained a little more. One general problem facing free text searches is that of different word endings. You may want all photographs and records referring to 'marriage', but if the query merely looks for the full string "marriage", it is likely that you would miss a great deal of valuable material where the word used is closely related, but not identical, for instance "marriages", "married", "marriageable", "marrying", "marry" and so on.

In order to overcome this problem, the Naga system uses a 'suffix-stripping' or 'stemming' algorithm, which is optional. This strips words down to their roots, in this case 'marri'. Since the letters 'i' and 'y' are interchangeable in this algorithm, this would find all the above variants of marriage, whether you typed in 'marry' , 'marriage' or any other variant in the query. (The full details of the algorithm which does this are explained in The paper by M.Porter, 'An algorithm for suffix stripping', **Program**, vol.14, no.3, pp.130-137, July 1980).

Although these can all be varied in the indexing specification, in the present implementation we have made certain choices as follows.

The program for suffix stripping and presenting words currently does the following.

'Free text' only fields:

Surrounding single quote marks are taken off: thus one can put foreign words such as 'arbre' or 'maison' in such quote marks in the text, but later find them by searching for arbre or maison.

The same is the case with brackets: thus one can put in (house) and later search for house.

The same is true of commas, full-stops etc. Thus if one types in a sentence of the form "daughter, father, and all the family." one can then search for daughter or father or family.

The same is true of hyphens; the hyphens are ignored and the two parts are each treated as a separate word. Thus for instance "double-banked" could be found as double or banked.

All words are 'suffix-stripped'. If you would like to see how a word appears when it has been suffix stripped, you can search for the word in the 'free text search' mode and then print out the 'terms' by which the record you have found is indexed. This gives the suffix-stripped version within single inverted commas.

Fields which have been indexed for both free text and structured queries.

The material is indexed in the same way for the "free text" mode. Thus, when you search in the "free text" mode, names, places etc. will be suffix-stripped and all surrounding materials ignored as described above.

These fields are also indexed for structured searching and

will appear in the structured query lists. In this form, there is no suffix stripping and all the surrounding characters are included.

Thus in the 'Structured' mode, (Harding) will be left as (Harding); likewise, punctuation marks attached to the words will be left there. Thus "Mary," will appear in the structured query as "Mary,". It is important to remember this in relation to persons, places, etc. In other words, in the structured form the whole string of characters, **including spaces**, is treated as one word. Thus if one put in 'Harding's wife', it would appear in the structured form as exactly that, 'Harding's wife'.

The effects and limitations on forcing upper and lower case.

One commonly puts in text in a mixture of upper and lower case, that is capital letters and ordinary letters (for instance, a name as Harding). In order to increase efficiency in searching, it is assumed that all text put in for free text searching will be 'forced' into lower case. This means that you can type in Harding, HARDING, harding or any combination of upper and lower case letters, and you will find every occurrence of Harding.

If you are indexing a field only for free text searching, it is therefore impossible to 'force' upper case, since it goes against the approach outlined in the previous paragraph.

On the other hand, it may be the case that for the structured mode, when lists of terms are presented to the user, such a strategy is not satisfactory. Firstly, it would be necessary to look for every separate type of string - hence one would have to look for Harding, HARDING, harding etc. on the list. Secondly, such a list would be unnecessarily long and untidy.

If you are only indexing a field for structured searching, it will be forced into upper case. This does not change the actual data when you finally see the record on the screen. It does mean that you will find records better, and have a more comfortable screen to look through.

The treatment of numbers or digits

In the free text mode, these are ignored entirely. Thus, for instance, if you type in 1990 it would not be indexed; 12house would be indexed as house; household12 would be indexed as household; house12hold would be indexed as two separate words, house and hold, the numbers being treated as a space.

In the structured mode, numbers are treated as a string of characters and can be searched for in the normal way: for in-

stance, if you type in 1990, you can search for 1990.

Words that are never indexed; the 'stop list'

A 'stop list' of words that are never indexed is included in one of the indexing specifications. This includes all one letter words, and the following other words:

'about' 'after' 'again' 'against' 'all' 'an' 'and' 'ani' 'down' 'dur' 'each' 'except' 'few' 'first' 'for' 'from' 'into ''is' 'it' 'more' 'most' 'out' 'over' 'own' 'per' 'same' 'so' 'some' 'to' 'togeth' 'too' 'under' 'until' 'up' 'us' 'veri' 'was' 'ar' 'as' 'at' 'be' 'been' 'befor' 'below' 'between' 'both' 'but' 'by' 'can' 'further' 'had' 'ha' 'have' 'how' 'if' 'in' 'no' 'nor' 'not' 'of' 'off' 'on' 'onc' 'onli' 'onto' 'or' 'other' 'such' 'than' 'that' 'the' 'their' 'then' 'there' 'through' 'which'
'while' 'why' 'will' 'with'.

They are in single inverted commas to indicate that they are the 'suffix stripped versions; for instance, 'dur' is 'during', 'ha' is 'has' and in several cases the final 'y' has been changed to an i ('onli' is 'only' for instance).

APPENDIX H

THE MUSQUITO TEXT PROCESSING UTILITY By Michael Bryant

MUSQUITO is a utility for processing text files intended for CDSi input. Its purpose is to allow the identification of items to be coded separately within a body of coherent text, and their automatic extraction.

The program works through a file looking for angle-bracket delimiters (< >). Any text within angle bracket pairs is copied and appended to the end of the current record (identified by the end of record mark - #).

To allow the automatic allocation of codes to the extracted text there are two important processing conventions:-

1. If the first character after an opening angle bracket is an asterisk then this is treated as the start of a field code. All following characters up to the next space are interpreted as the code and are removed from the source text.

2. If the first character after an opening angle bracket is another opening angle bracket then it is assumed that the following text, up to the first closing angle bracket, is to be removed from the source text. This can be used for the automatic

coding of group codes, or the "on line" creation of extra material.

To allow a comfortable text entry style spaces on either side of the main angle brackets are reduced to one in reconstituting the basic text, and a trailing space after a muscat code or enclosed angle brackets is removed. See the conversion example below.

Using MUSQUITO

Load the program MUSQUITO.EXE into the directory which will contain the files to be processed, or locate it in the DOS search path.

The program can be used with command-line parameters or interactively. With command-line parameters the program processes the given file, then finishes. When used interactively you are asked whether you wish to process further files as each one finishes.

To run from the command line, log on to the directory containing the source file and type:-

MUSQUITO FILENAME_1 [FILENAME_2]

FILENAME_1 is the name of the source file for processing. It can have the following forms:-

NAME the program assumes the file has the default extension .MTO (file = NAME.MTO).

NAME. the file has no extension (file = NAME).

NAME.EXT the file is exactly as given (file = NAME.EXT).

FILENAME_2 is the (optional) name of the destination file. If there is no FILENAME_2 then the destination filename is set to the name of the source file with the extension .TXT (file = NAME.TXT), otherwise the rules for FILENAME_1 are followed with .TXT as the default extension.

Examples

MUSQUITO GURUNG
processes GURUNG.MTO to GURUNG.TXT

MUSQUITO GURUNG.

processes GURUNG to GURUNG.TXT

MUSQUITO GURUNG.ABC GU5

processes GURUNG.ABC to GU5.TXT

```
MUSQUITO GURUNG.TXT GU5.ZZZ
```

processes GURUNG.TXT to GU5.ZZZ

If source and destination filenames end up the same then a warning is given and the destination filename is changed by appending (or altering the last character to) a dollar sign.

```
MUSQUITO GURUNG. GURUNG.
```

processes GURUNG TO GURUNG$

```
MUSQUITO GURUNG.TXT
```

processes GURUNG.TXT to GURUNG.TX$

To run interactively, simply type MUSQUITO and the program will start, prompting you for filenames as required. The same assumptions apply as with command-line parameters.

Limitations

MUSQUITO works with files in the currently logged directory. You cannot give directory paths as filename parameters.

Any size of file can be processed as long as there is room for the destination file on the disk you are using, and there is enough memory available to hold the copied material from any one record.

Most error conditions generate messages and close the program in an elegant fashion.

Example Conversion

Source File = EXAMPLE.MTO:-

```
*rec This is an example of the use of angle brackets as delimiters for MUSQUITO. <*name MUSQUITO> is a utility for processing files intended for <*prog Muscat> input. Its function is to allow the <*action marking up> of Muscat fields within other fields, especially long texts.
#
*rec A feature is the use of angle brackets within angle brackets. The purpose here is to allow <<*prog *specific> group code> specifications.
#
```

```
Destination File = EXAMPLE.TXT

*rec This is an example of the use of angle brackets as delimit-
ers for MUSQUITO. MUSQUITO is a utility for processing files
intended for Muscat input. Its function is to allow the marking
up of Muscat fields within other fields, especially long texts.

*name MUSQUITO
*prog Muscat
*action marking up
#

*rec A feature is the use of angle brackets within angle brack-
ets. The purpose here is to allow group code specifications.

*prog *specific group code
#
```

APPENDIX I

FILES AND MACROS

Introduction

The system is contained in a set of programs and macros, some of which can be changed, some not. There are four 'executable' files. Three of these are kept either in your route or in a directory (e.g. dos) on your normal path so that they can be accessed from anywhere. These are:

muscat.exe - the main program which is invoked when 'muscat' is called
cdsi.exe - the program to run the Cambridge Database System Interactive
musquito.exe - the program to run a text-editing utility

There is another executable file:

discat.exe - the program to run discat (disc cataloguing system)

This is kept in the sub-directory \muscat\discat, along with 'overlay', which are self-contained programs forming a 'runner' which makes it possible to use a DA database without other software.

The 'macros' or small programs turn the general

system 'Muscat' into a particular application useful for specific purposes. By modifying these macros it is possible to develop and change the system and application. Since the macros are held as text files in various directories, it is not too difficult to alter them using a word processor. When they are altered, the next time that macro is used, the new version will be implemented. (A way of automatically re-writing some of the simpler macros is provided in CDS Interactive, as explained in the separate manual).

In order to alter and modify the macros, it is necessary to understand how they work in general, and what each one looks like in particular. This appendix sets out to explain how this is done and gives the function of each of the specialised macros. The full text of the macros can be seen by printing them out as needed.

A general description of macro commands is given in the Muscat Manual, 3rd edn., pp.21-2. It is explained that the name of a macro command must always begin with a letter and hyphen. Thus, in our case, the major macros start c- , for instance c-build. This tells the computer to use the macro called 'build' which has been set up using an 'alias' c. All the 'macros' in the directory \muscat\macros\cds will thus be invoked using c-.

File names.

Within Muscat, the following extensions are associated with the following types of Muscat file:

type of file	extension
text file	txt
muscat sequential file	mus
database file (DB or DA)	da
pag output	pag

In a muscat command, the appropriate extension is added on to a file name if no extension is supplied by the user. Thus

```
c-build mills to mills
```

is equivalent to

```
c-build mills.txt to mills.mus
```

To refer to a file which has no extension, the file name should be terminated with '.', which tells Muscat that the

extension is null, for example

```
      c-build mills. to recs.
```

This builds from the file 'mills' to the file 'recs' with no extension.

The over-all structure of the files and macros.

The general structure of the files in CDS may be represented best in a hierarchical way as follows:

C: (the root directory)

```
MUSCAT - FORMATS        - WORK
       - MODS
       - DELBASE
       - CDS
       - DISCAT
       - CDSI
       - MACROS - UT
                - EL
                - DISCATEL
                - CDS
                - CDSI
```

THE MUSCAT DIRECTORY

This contains the whole system, apart from the executable files. There are two small directories, with very little in them.

Formats: this has two empty directories and formats for 'el' (i.e. muscatel) and for 'cds' (i.e Cambridge Database System) and discatel (elementary discat system).

Work: just has two empty directories. Into this will go temporary work files while the system is in use. It needs to be cleaned out periodically by deleting **all** the files in it.

MUSCAT\MODS

The main set of programs is held in the sub-directory 'Mods'. This holds all the major Muscat commands, such as 'build', 'checkid', 'dbcreate', 'number' etc. There are 103 files in all, which may be seen by examining the 'Mods' directory. They cannot be altered as they all have the

extension .cin.

MUSCAT\DELBASE

This directory, which can be moved from within the 'Muscat' directory to a higher level (as a sub-directory of the root directory), contains an example of the 'Discatel' (or elementary discat) system. This allows you to set up your own database with its own format, record structure, indexing requirements and print specification. It contains the following macros and data.

```
sample.txt   - a small sample of records to try out the system
pspec.txt    - a specification of how the material is printed
pspec1.txt   - an alternative specification for printing
pspec2.txt   - an alternative specification for printing
intro.txt    - the introductory screen pages for a database
db.da        - the current database, if there is one
dspec.txt    - the screen specification
ispec.txt    - how a document is to be indexed
iexp.txt     - how a document is to be indexed
introduc.txt - a longer version of the introductory screen pages
```

A fuller description of these, with a full print-out, is contained in Appendix O.

MUSCAT\CDS

It is in this directory in which any files which are to be used in he system (for instance text files which are to be built and put into a database) should be kept. The different extensions (as explained at the start of this appendix) will show what kind of file or macro is being referred to.

This directory will contain any database files. An updatable database (DB) will be called db.da, a direct access database (DA) will be held in two files called darec.da and daterm.da.

The directory contains some temporary work files, called s.mus, r.mus, w1.mus and w2.mus. If these become too large, they can be deleted.

It may also have a format.mus file, to help define a format for the data, which should not be deleted.

It also contains specifications of what the data should look like when printed out on the screen using the full system, with the numbers of records on (pspec.txt), with the numbers of the records off (off.txt), and in 'q' and 'print'.

It contains an initialisation sequence, a way of going into the database called init.txt.

The other major file is called intro.txt. This contains all the special introductory materials, for instance the help pages, the pages of choices etc. It can be modified to suit your needs, being a text file, and then re-loaded, as explained in Appendix K. It is automatically loaded in each time a new database is set up, because the macro for setting up a database, c-create, includes a command asking for intro.txt to be added into the database.

MUSCAT\DISCAT

This directory contains a completely self-contained system, which allows you to run a Database (DA) without any of the other software, as described in the 'readme' in that directory. The files are as follows:

overlay - a program to run the system
create.bat - used in running the system
readme - how to enter the system
format.mus - the format for the system
daterm.da - a small trial database
darec.da - as above, the complementary records
pspec.txt - the print specification
start.txt - to start the system with a mono screen
start2.txt - as above, with a videodisc
start3.txt - to start the system with a colour screen
start4.txt - as above with a videodisc
discat.exe - a program to run the system

MUSCAT\CDSI

These are the macros which are used by the CDS Interactive software to set up particular applications, as explained in Appendix V.

MUSCAT\MACROS

This sub-directory has four sub-directories within it.

THE UT MACROS

One is called ut, short for utilities. This has a number of files or macros which can be edited since they are text files (with the extension .txt). The macros in this are as follows: dump, help, init, merge, select, sort, paj. They can be invoked by using a command starting with the prefix u-, for instance u-select. The purpose of these various utilities is explained

in appendix F, under 'Command set u'.

THE EL MACROS

This is the 'Muscatel', or elementary Muscat system, as described more fully in Appendix A of the Muscat Manual, where the full text of each macro is printed. It contains the following macros (all of which can be modified to suit your needs): build, codemap, combine, cycle, init, kill, list0, list1, makef, merge, mult0, mult1, number, print, reduce, ret, scramble, sort, unlev, update, all, asort, help, j1, print2, sort2, oldform.

This set of macros can be accessed in two ways. You can either go into it by entering Muscat with the command 'muscat sys el', or by going 'u-select el'. It is much easier to adapt and use than the full Muscat system, and is particularly useful for numbering, sorting and printing documents. The full text of these macros is printed on pp.167-171 of the Muscat Manual (3rd edn.).

Once inside the muscatel system, the commands are preceded with the prefix g-, thus g-build, or g-number.

The default muscatel system is designed around 26 string fields called *a, *b...Sometimes users need more than 26 fields, or need fields with more suggestive names. To achieve this, you can define your own format and use it to override the Muscatel format. How to do this is explained in the Muscat Manual, p.189.

THE DISCATEL MACROS

This contains a special set of macros which work with the 'Discatel' system. A number of these overlap with those in the El (Muscatel) system. Those which vary from Muscatel, are printed in Appendix S.

THE CDS MACROS

This sub-directory contains the main set of macros which, when used with the prefix c-, allow you to set up a database system and interrogate it. A functional listing of these may be given as follows:

Setting up a format for the data.

makef - provides a format or template for the records.

Helping to organise and clean the data.

dateind - to sort a batch of records by the *prod *d field

klind - for linking maps with synonyms (see klspec for a
description of this)

kdind - to sort a batch of records by *kd field

sort - to sort a large file (equivalent to u-sort)

voc0 - to produce a diagnostic output of everything except the *t
and *ns fields

ied1 - for moving data from a group field (*g) to an identity field (*i)

bk1 - creates a raw contents list for a book

kill - kills (suppresses) a field or fields out of a record

acqr - sorts by the reference (*r) in acquisition field

lbrief - to print out only the short caption (*u) and identity (*i) fields

Creating appropriate text records and formats

bksplit - splits books/manuscripts into a text and record part

liste1 - creates appropriate bracketing for references

locmap - creates variants and standards names in location records

Checking the records and numbering them.

checkid - checks the identities of the records

check0 - checks that certain main fields (*c *t *i) are in a record

number - automatically numbers the records

termsof - if a single record, prints out all the terms of a built record as it would be indexed in the database

Building the records and unbuilding them (printing).

build - to build into 'muscat' records for entry to database

list - to list out the built records from muscat files

print - to print out the built records

dump - version of print which shows the hierarchical structure of the fields

listm - lists out (prints) all 'marked' files (suitable for editing)

Setting up the database

create - sets up a DB database (prompting for size), called db.da

setupir - sets up a DA database (from a built file)

Indexing records and adding them to the database.

index - specifies which fields to index and words to leave out

add - adding records to a database

del - deleting a single record from the database

update - adding records to a database which have already been added in before and only slightly changed

batchadd - to batch add a file, running build/liste1/number/add

Going into the database.

dbsys - going into the DB database system

disc - going into the DA database system

q - going into the 'q' or query database system

Printing the results.

pspec - the print specification for the screen version of cds

dspec - the print specification for 'q' and 'print'

Saving and printing files of marked records.

getmrecs - to turn a marked file into a muscat file in a DB system

getdiscm - to turn a marked file into a muscat file in a DA system

getrecs - searching for a set of records outside a database system

Setting up the function keys.

The function keys can be set up to run macros. When in CDS 2000,
it is possible to do various things by pressing the function keys
in combination with control, as follows:

f1 - saving a record to a temporary file within the Database

f7 - printing out a hard copy of the current record

f8 - going out of the database temporarily into Muscat, or ending an edit entered through f1 above

THE CDSI MACROS

This sub-directory contains specific macros used in the CDS Interactive system (as described in Appendix V).

A note about macros; and how to stop captions being truncated.

Macros specify various parameters. Some of these are machine dependent, and come after the word 'with'. Hence, 'with v' is a machine-dependent command to link the machine to a videodisc.

Other controls are machine-independent, and are prefixed by the word 'opts'. Thus one can specify the number of records to be seen by typing 'opts z2000'.

It is thus important when changing a macro, to put the change in the right place. An illustration will also allow you to make a change. In the macro called dbsys.txt, which takes you into the database, there are a series of options defined after <opts>. Just before this, you will see the line with just 'm69' on it. This has the effect of setting the margin for caption lists to 69 characters. We have found this useful, since the purpose of these caption lists is to be able to look very quickly through a large number of records, and usually a line is enough to give one the idea of whether you want to see the full record. It does the same with the 'marked' files.

If, however, your captions were pretty short in any case, seldom exceeding, say, two lines, and you wanted to see the whole of them, then this instruction could be deleted. Then the whole of the caption would appear. Since our captions were often several lines long, we found a number of difficulties in printing on the screen without this truncation. So you should be wary if you decide to remove 'm69'.

APPENDIX J

SOME EXAMPLES OF EDITED RECORDS WITH THEIR CODES

Record indexing a sequence of moving film

```
*c 16mm colour film taken by Ursula Graham
Bower between 1940 and 1944
*m films
*u fish poisoning expedition at river near Hangrum
*m films *ke Zemi *prod *p Graham Bower /Ursula
*d 11.1940 *acq *p Pitt Rivers Museum Archive, Oxford
*g *i B.11832=11834 *u long shots of procession
walking to river
*g *i B.11835=11838 *u close-up of people carrying
vegetation
*g *i F.11839=12039 *u beating poison into river
*g *i F.12042=12219 *u man beating fibre
*g *i F.12220=12467 *u little boys searching for fish
*g *i F.12470=12628 *u little boy searching for fish
*g *i F.12631=12700 *u little boys searching for fish
*g *i F.12701=12899 *u men beating fibre on rocks
*g *i F.12900=13135 *u men scrambling up river looking
for fish
*g *i F.13138=13302 *u crowd of men swimming down
river looking for fish
*qv black and white photographs taken by Ursula
Graham Bower {\B.52911\} {\B.52912=52915\}
```

#

Record describing a museum artefact

```
*c colour photographs of Naga artefacts from
various sources
*m artefact
*i B.642 *u Hat of cane-work with two [missing]
black feathers and a large boar's tusk. A
chaplet of long pig's bristles encircles the hat
interspersed with sharp bristles dyed red. An ornament of
plaited yellow cane or orchid-stem is in front attached
round the 'chaplet' cane foundation. Phom.
*m artefacts *ke Phom *z 12cm (height of cap)
*prod *e Phom *prod *r 4:218 *coll *p Hutton /J.H.
*acq gift *p Pitt Rivers Museum, Oxford
*d 1919 *r Hutton I.183
*ns descriptions derived from original source material
unless in square brackets or otherwise stated
#
```

Record describing a photograph

```
*c black and white photographs taken by C.R.Stonor,1946-1948
*i B.51668
*u gate at Chepoketami village
*m photographs
*kl Chepoketami
*prod *f celluloid negative *prod *p Stonor/ C.R.
*prod *d 5.1946
*acq *p Pitt Rivers Museum Archive, Oxford
*acq *r 526
*acq *n documentation based on index in Pitt Rivers Museum
Archive
*ns text in square brackets, if any, is taken from illustrations
in Stonor's article, 'Feasts of Merit among the Northern
Sangtam' #
```

Record describing a sound recording

```
*c wax recording, originally on wax cylinders
*m sound
*i V.46050=49750 *u "Lipeli", a kind of song sung
by Semas when working in the fields *m sound *t ["Lipeli"
or "Lephile", sung when reaping.] *ke Sema
*prod *p Hutton /J.H. *n information derived from source
unless otherwise indicated
*prod *d 1919
```

```
*acq *p Pitt Rivers Museum Archive, Oxford *r cylinder
no.9 *qv J.H.Hutton, Sema Nagas, 1921, 115-6
*ns sound recording made by anthropologist #
```

Record describing a location and map

```
*u Place name: Wangla *t map: {\C.47160\} map record: {R.64160\]
*ke Konyak *kl Yangla = Wangla *kl Lakma = Wangla
*ns a location and map record #
```

Record describing an extract from a diary

```
*c Christoph von Furer-Haimendorf,
Naga diary 5, translated by Dr Ruth Barnes
*m diary
*u physical anthropology, colour of hair *kp Ngapnun
*ke Konyak *kl Longkhai *kd 11.4.1937
*prod *p Furer-Haimendorf *d 1.4.1937-26.6.1937
*t In the meantime it also was surprisingly easy to find
people for Kauffmann to measure {\B.51109\}. The Ang started
it off. Others followed and finally we could even persuade
some of the women to allow themselves to be measured.
Remarkable for many is the light and plain hair which has
almost blond strands and is not tight. The mongoloid fold
is especially developed among some, for example Ngapnun.
Unusual is also the slender and graceful build which is
expressed in hands and feet as well.
*ns anthropologist's fieldwork diary #
```

Record describing an entry from a fieldwork notebook

```
*c Christoph von Furer-Haimendorf notebook 3
*m notebook
*u were-tigers of Kongan and his powers; shamans *ke Konyak *kl
Kongan *kd 26.8.1936
*prod *p Furer-Haimendorf *d 8.1936-6.1937
*t In Kongan is a real thibu who is also a were-tiger,
Lem-ang, who is also gaonbura. A time ago the
Mohore of Merankong (an Ao) slept at night in the jungle
between Tamlu and Namsang. Suddenly they saw a tiger and
fired at him without hitting him. The tiger (172) ran
away. When they came to Borjan they met Lem-ang and the
first thing he said: "Why did you try to shoot me, I
hardly could escape?" Lemang always knows in what village
and what house illness is and if it is a man or a woman.
He goes f.i. as tiger to Tamlu. When he is called to a
sick man first his spirit goes to the man and feels on his
```

```
chest and body where the illness is.
*ns anthropologist's field notebook
#
```

Two records from a book

```
*c Pride and Prejudice
*m book
*prod *d 1798
*kp Bennet/ Mrs
*prod *p Austen/ Jane
*u marriage and fortune
*t  It is a truth universally acknowledged,  that a single man
in
possession of a good fortune must be in want of a wife. #

*c Pride and Prejudice
*m book
*prod *d 1798
*kp Bennet/ Mrs
*prod *p Austen/ Jane
*u marriage strategies of neighbours
*t  However little known the feelings or views of such a man
may be on his first entering a neighbourhood,  this truth is so
well fixed  in  the  minds of the surrounding  families,  that
he  is considered  as the rightful property of some one or other
of their daughters."
#
```

Record describing an ecclesiastical court case

```
*c Archdeacon of Colchester's Court
*prod *d 17.1.1581
*kp Green/ Mary
*kl Earls Colne * Coggeshall
*acq *r Essex Record Office, D/ACA/8 fol.238v
*u Office of the Judge against Mary Green of Earls Colne: she
appeared personally and it was objected against her by the
churchwardens of Colne that she is vehemently suspected of sor-
cery and witchcraft etc. Therefore the judge ordered her to
purge herself under the hands of four of her neighbours, of
three were of Coggeshall and one of Colne aforesaid, at the
next court.
*ns Ecclesiastical court record #
```

Records describing some parish register entries

```
*c Parish Registers, Burials #m a
*acq *r Essex Record Office, D/P209/1/1 #m b
```

```
*kl Earls Colne #m c
*k burial #m d
#l a b c d
*kd 5.7.1567
*kp Allen/ Thomas  * Allen/William <father
*k son #
*kp Abbott/ Joan * Abbott/ Henr <father
*k daughter #

*kd 13.11.1567
*kp Ells/ Rose * Ells Jeffrey <father
*k daughter #
*kd 4.2.1568
*kp Read/ Margery * Read/ Robert <father
*k daughter #
```

Some records of book titles

```
*kp Ahern/ Emily Martin
*kd 1972
*kl Cambridge University Press * Cambridge
*z ix, 144 pages
*m book
*acq *r 495.200
*u Chinese Ritual and Politics #

*kp Burling/ Robbins
*kd 1968
*kl Pennsylvania University Press
*z 377 pages
*m book
*acq *r 368.02
*u Rengsanggri ; family and kinship in a Garo village #
*kp Smout/ T.C
*kd 1973
*kl William Collins * London
*z 540 pages
*m book
*acq *r 543.27
*u A History of the Scottish People 1560-1830 #
```

Two archaeological records

```
*c Late Pleistocene Fauna
*kl Inamgaon * Maharastra
*kd 1968-1983
*prod *p Dhavalikar/ M.K. * Sankalia/ H.D. * Ansari/ Z.D.
*acq *r INM/134/DC
```

```
*qv Fig. 5.32.2
*k Femur * proximal right
*u The specimen is a proximal end of femur preserved with head,
fovea capitis, trochanter major, trochanter minor and fossa
capitis. The specimen is mildly rolled and has, therefore,
obliterated edges.
*ns archaeological record #

*c Late Pleistocene Fauna
*kl Inamgaon * Maharastra
*kd 1968-1983
*prod *p Dhavalikar/ M.K. * Sankalia/ H.D. * Ansari/ Z.D.
*acq *r INM/63/DC
*k Tibia * condyler epiphysis * right
*u The specimen is represented by condyler epiphysis.
Inter-condylar eminence is well pronounced. Fossa of sulcus
muscularis is well represented with the edge of condylus
lateralis. Tuberositas is also well preserved in the specimen.
*ns archaeological record
#
```

APPENDIX K

INTRODUCTORY CHOICE AND HELP PAGES

The interface between the database materials and the user can be modified to suit your needs. An exercise to show how this is done is included in tutorial one below. Here there is part of the general text of the current introductory and help pages. It should be noted that the records to specify pages of year dates have not been included here, apart from one small sample to show their nature. Likewise all but one of the 'Help' pages has been omitted. If you want to modify these and need the full text please look at the actual text in \muscat\macros\cds\intro.txt.

The introduction and help pages appear as a number of control (A) records, help (H) records or text (T) records. These can be added in either as part of intro.txt, or as single records. The (abridged) text is as follows:

```
*i A.1
*g1 Welcome to the Cambridge Database System
{g1} Press '\vH\n' for Help:    {u 34 1 \H.1\=h}
{g0} \vI\nntroductory text:     {u 34 1 \T.10\=i}
{g0} \vT\nutorials:             {u 34 1 \T.20\=t}
{g0} \vC\nontents:              {u 34 1 \T.30\=c}
{g0} \vF\nree text query:       {u 34 1 \A.0\=f"sl"}
{g0} \vS\ntructured query:      {u 34 1 \A.2\=s}
```

```
{g0} \vB\noth free text with structured: {u 34 1 \A.2\=b"sl"}
#


*i A.2
*g1

Select one or more of the following:

{g1}\vY\near                {u21 1\A.4\=y}
{g0}\vO\nther date                {u21 1:1=o">enter as yyyy or
yyyy/mm or yyyy/mm/dd"}
{g1}\vP\nerson involved     {u21 1:2=p">enter surname>lc"}
{g1}\vL\nocality name       {u21 1:3=l">enter locality name>lc"}
{g1}\vE\nthnographic group  {u21 1:4=e">enter group name>lc"}
{g1}\vM\nedium of recording {u21 1\A.3\=m}
{g1}\vS\nource of material  {u21 1:7=s">enter name>lc"}
{g1}\vV\nideodisc frames  {u21 1:6=v">enter a frame number"}

#

*i A.3
*g1
MEDIUM OF RECORDING
{g0:5"sketches"} sketch
{g0:5"films"}   film
{g0:5"photographs"}  photograph
{g0:5"sound"}  sound
{g0:5"artefacts"} artefact
{g0:5"maps"}    map
#

*i A.4
*g1
Please select from among the following date ranges:
{g1}Up to 1879    {u16 0\A.5\}
{g0}1880 - 1919   {u16 0\A.6\}
{g0}1920 - 1959   {u16 0\A.7\}
{g0}1960 or after {u16 0\A.8\}
#

*i A.5
*g1
{g1} 1847:{:1"1847"}
1855:{:1"1855"}
1874:{:1"1874"}
1875:{:1"1875"}
1876:{:1"1876"}
```

```
{g0} 1877:{:1"1877"}
1878:{:1"1878"}
1879:{:1"1879"}
#

*i T.19
*g1 COPYRIGHT

*g1 Copyright Cambridge Experimental Videodisc Project,1989
*g1  The Project encourages educational and research uses of
the material on the videodisc and computer disc. Copyright in
some of the  material,  however, is held by  particular
individuals  and institutions (see Acknowledgements). Please
consult the  Project, therefore, if such use entails copying
onto other media. *g1  The  visual  material  on  the
videodisc  can  be  used independently or in conjunction
with the  indexes,  texts  and associated retrieval software
produced by the project.
*g1 The computer software, the Cambridge Database System (c),
is copyright. *g1  All  enquiries for purchase or licences to
use the  data  or software without a videodisc system should be
sent to: Rivers  Video Project, Dept. of Social Anthropology,
Free  School Lane, Cambridge CB2 3RF

*g1  All  enquiries for purchase or licences to use the  data
or software  with  an  interactive videodisc  should be made to:
Cambridge  Interactive,  Barnwell  House,  Barnwell  Drive,
Cambridge CB5 8UJ (O223-214893). #

*i T.30
*g1 TABLE OF CONTENTS
*g1 {\T.31\} By name of collector or author
*g1 {\T.41\} By medium or source
#

*i A.98
*g1 \v 'Full record retrieval'\n {u10 l+10} to retrieve
 by the records.{l-10}
*g1 \v 'Data retrieval'\n {u10 l+10} to retrieve by the data
(images
or text). {l-10}
*g1 \v 'Caption retrieval'\n {u10 l+10} to retrieve by a list of
short
captions to the records.
#
```

APPENDIX L. HOW THE VARIOUS FIELDS ARE INDEXED; A SUMMARY

While it is recognised that the particular fields which have been described in chapter two will only partially satisfy some

users. This appendix will explain how the full system described in the manual works. This will allow you to edit the macros. A simpler, automatic, way of setting up indexing and other specifications is described in the CDS Interactive manual.

Fields that are not indexed at all

At present, the following fields are not indexed at all in the system (i.e. terms from these fields are not put into the index).

*t - longer text
*ns - notes
*qv - see also
*z - size
*r - archival references
*f - form or function

These can hold letters, numbers, or a mixture of the two and are treated as 'string' fields.

Fields that are indexed for free text but not structured queries.

At present the following fields are indexed so that you can find their contents through free text queries, but do not appear in the table of 'structured' queries. (Structured queries can, in fact, be made on these terms in a rather more complex way by using 'q' or 'embedded queries, as explained elsewhere.)

*u - short description
*k - keywords

Fields that are indexed for free text queries, but not for 'relevance feedback'.

In this case the terms are not included in working out and presenting relevance feedback and query expansion. The purpose of this category is to deal with information which you may want to search for, but would be unhelpful when using query expansion. At present this applies only to the 'title' field, *c. It may be that you would like to look for all 'fieldnotes', as in the title. But it only obscures query expansion to have these terms included in the query expansion algorithm.

Fields that are indexed for free text queries and also for structured queries.

These fields are indexed once, but are accessible in two ways, through the free text system and through the structured query page on the screen. They divide into five sub-categories:

i. A person or *kp field. This allows you to give a forename, surname and detail (e.g.Mr).

ii. An 'ethnic group' or *ke field. This allows you to have a main part and a detail, of the form Angami <Eastern. The possibility of groups, with sub-parts, may be useful.

iii. A 'location' or *kl field. This deals with geographical places and areas and may again have a main part and a detail, as with ethnic group.

iv. A 'date' or *kd field. This expects a string in the form of some numbers, as described in chapter 2.

v) A 'medium' or *m field. This we use to indicated whether something is a photograph, film, map or whatever.

Fields that contain fields.

We also have three fields which can contain other fields. These are as follows:

*acq - an acquisition field
*prod - a production field
*coll - a collector field

These could contain within them certain sub-fields as follows:

*f - function or form
*p - person
*d - date
*e - ethnic group *l - location
*r - reference number
*n - note

Thus you could have *acq *p, meaning the person who acquired something. This is a potentially useful category of fields. All the fields *p *d *e *l are indexed for free text searching, under all three of the categories. The one exception is *acq *p which, often describing a museum or archive which has acquired an artefact, has been dealt with differently. It is not indexed for free text searching, but appears in the structured query choices as 'Source of material'. The *f *r *n fields are not indexed at all.

Group fields.

A explained in Chapter 2, there are occasions when a group field (*g) is very useful. At present the group field can contain the following sub-fields, u m t k kp ke kl z n.

Integer fields

As explained in more detail elsewhere, integer fields are fields which can only contains integers or numbers and are fields upon which mathematical calculations can be made. We have not had any use for such fields.

Total list of permitted fields.

How fields are treated	Codes
Not indexed at all	t n qv z r
Indexed for free text queries	u k
Free text, not query expansion	c
Both free text and structured	
'person' type	kp
'group' type	ke
'location' type	kl
'date' type	kd
'medium' type	m
Fields that contain fields	acq prod coll
'date' or 'location' type	d l
'person' type sub-field	p
Integer fields	-

APPENDIX M

CROSS REFERENCES BETWEEN RECORDS.

Introduction; embedded actions.

It is possible to move between records in various ways, as well as merely cross-referring to them by a *qv as indicated above. In essence it is possible to embed a cross- reference to another record in the database. This is done by placing it within curly brackets, { }, within the record as follows:
some textual materials... { embedded action } .....

This will allow the user to make a choice when reaching the portion in { } brackets.

The three kinds of cross-reference or embedded actions.

When a record is printed, a cross-reference of this kind appears on the screen as a selectable box (or 'icon'). When this is selected the record which is cross-referred to is shown. How it is shown, and subsequent actions, are determined by the use of different kinds of slashes placed within the curly brackets.

A passing cross-reference to another record.

In the case of back-slashes, \ \ , selecting the box will take you to that record. A subsequent selection of 'R' or Return takes you back to the record from which the original selection was made. (Thus it is like a subroutine call in computer programs).

For instance, supposing you were looking at a diary entry about a dance. The diarist might mention that he had taken a photograph of the dance and you might want to be given the opportunity to see the photograph and then return to the text. The photograph in question is at frame 2000 on the videodisc, and being a still frame is coded as B.2000.

The way to allow you to look at this would be to insert:

"Then I took a photograph of the dance {\B.2000\}. It
was a beautiful sight...."

Having seen the photograph, you return to the text.

A cross-reference taking you to a new record.

If you were to use forward slashes, / /, you would be taken to another record or picture. But on selecting R or Return you would not go back to the original record. It is thus a form of jump or 'Goto'. Thus if you had the following:

(record 25).....some text...{/T.2000/}.....more text...

On selecting the box you would go to text record 2000. But on pressing 'Return' you would return to the record associated with text 2000, rather than the original record 25. In fact this form of cross-reference cannot be used in 'R-records', the normal indexed records.

Cross-references to certain menu selection possibilities.

The third type of cross reference, with vertical slashes, or |, is rather different. Only one cross-reference of this form should appear in a record. Its presence causes certain menu selection possibilities to appear on the menu bar at the bottom of the screen. it takes the form:

{|A.200|}

An important example of its use is as follows.

Cross-references between one page of text and the next.

If you have a page of text, you will need a way to look at the next or previous pages, or to go back to a table of contents. For instance, you may be reading a diary. You have on the screen a certain day and want to go to the next or previous days. Or you may want to go back up to the page of contents. This can be done using the vertical bar cross references.

At the end of a page of text you can type:

{|T.u T.p T.n|} where u,p and n are integers or numbers. This will give rise to the menu bar items UP PREVious and NEXT. Selection of one of these will take you up to the higher level of the table of contents, back to the previous page of the diary, or on to the next text, as specified.

Supposing you have a diary whose table of contents is numbered as text record 100, i.e. T.100, and which has three entries numbered T.200, T.201, T.202. Supposing further that you are at the entry numbered T.2O1 and want to be able to move around the diary. You could put at the end of the text record number T.2O1 the following:

{|T.100 T.200 T.202|}

That would enable you to move up, back and forward. The method for setting this automatically is explained in exercise of Tutorial 1 in Part C. An item in the selection above may be suppressed by using 0 or zero. For example:

{|0 T.200 0|}

would give a selection possibility for a previous page, but no choice for up or next.

Cross-references from 'R' records.

Elsewhere it is explained that R-records are the indexable items in the database. A retrieval request will retrieve R-records and only R-records. An R-record is a complete item of information in itself, or it may refer to further information in the system. In the latter case you should again use the {|...|} cross-reference. For instance:

{|T.200|} - would take you across to a text record

{|B.2000|} - is a reference to a still image

When this is put in, a menu item, SHOW will appear at the bottom of the computer screen. Selection of this will take you to the cross-referred item, whether text or image. On pressing R or Return, you will return to the R-record from which one has come.

Printing specifications within cross-references.

It may be the case that you will want to specify how something to which you cross-refer is to appear on the screen. This is possible by using 'print' commands within the { } or curly brackets. An example of what would be typed is as follows:

some text....{g1 u20 1 \T.1217\}....some text.....

This would take you to text record 1217, and lay it out on the screen by obeying the various commands g1 u20 1. These simple layout controls are as follows.

Some print layout controls.

It is possible to put print controls into the A, T and H records. Examples of their use will be found in Appendix K. These print controls should be put inside { } brackets. Thus, to move to the next page, you would type {n}. As will be seen in Appendix K, it is also possible to use *g1 and *g0 in certain circumstances to create new lines. The main print layout controls are as follows:

g N' - output N blank lines (for instance g1 will output one blank line)

l=n (r=n) - set the left (or right) margin to n

l+n (r+n) - increase the left (or right) margin by n

l-n (r-n) - decrease the left (or right) margin by n

d n - go to the next page if within n lines of the bottom

n - newpage

s - space

t m n - tab to position m leaving an n space gap

u m n - as above, but ignoring the left margin setting

p c - output punctuation character c

c < - set right justify position

> n - right justify to n

v n vertical drop to n

Ranges of non-textual items.

It is possible to specify ranges. This is done by giving the start and end number, separated by "=" or an equals sign. Thus you could have the following example:

B.100=108 - meaning photographs 100 to 108

Thus a cross-reference of the form {\B.101=108\} would, if selected, take you to a set of photographs, which could be looked at one after another.

In the same way you can refer to moving film, as follows:

F.1000=1500 - moving film, from frame
number 1000 to frame number 1500

In this case, on selecting some film, the first frame of the film would be shown. You can then use controls to play it backwards and forwards, at ordinary speed or in slow motion.

It is possible to combine these in a list, for example:

some text...{\B.216 * B.387=390 * F.2189=3000 * B.22\}...text..

This would allow you to look at a series of still and moving images. These lists must not contain R. T. or A. records, but just images on the videodisc.

Associating a string of text with a cross-reference.

It is possible to associate a piece of text with a

cross-reference by putting a string ('....' or "...." or /..../) after the cross-reference.

An example would be : {\A.14\ ">Name of author"}

This would mean that ">Name of author" would appear on the screen after the box. This enables you to provide guidance on what is being asked. An actual example of its use will appear later.

Cross-references to programs outside the database.

There is also a special method of invoking commands outside the CDS 2000 system. This is done by using the \X.0\ reference, followed by the commands within inverted commas. For instance, if you wanted some MSDOS commands to be obeyed, you could insert in a record:

{\X.0\ "MSDOS XCOPY c:\jo*.txt a:"}

The command within the inverted commas would be obeyed. Upon completion of the external command, control returns to CDS 2000 at precisely the point from which the command was invoked.

APPENDIX N

HOW TO MAKE AN EMBEDDED QUERY

How to make an embedded query, using 'Q' records.

A powerful feature of the system is the possibility of adding an 'embedded query' from anywhere in the system. Thus you could be in a record and allow a user to run a sub-query from that record by pressing on a box, and then return to the record. One major use of this, for instance, is in setting up a table of contents. A list of options can be presented, each one with a box. For example, you could have:

photographs taken by J.P.Mills
photographs taken by J.H.Hutton
photographs taken by W.G.Archer

and so on. Each would have a box against it holding an embedded query which you have set. On selecting this, the query would be executed and you would be able to see all the relevant photographs.

Embedded queries use a 'Q' (Query) record. They take the following form:

{\Q.n\">A>B>C>D"}

where

n is an integer or number
A,B,C,D are pieces of text
> is any character not in A,B,C,D (the sign > is a good one)

It is important not to end with a fifth >.
But you can leave out >, as in the following:

{\Q.n\">A>B"}

If, however, you leave out earlier text fields, for example B and C, then you must remember to put in the > for them, as follows:

{\Q.n\">A>>>D"}

This is the structure, the contents is the following:

n = the number of items to be retrieved, for instance you might want just to show the first 20 photographs, in which case this number can be set at 20. It is possible to set it at a higher number than the actual number of items; if you think there are about 340 photographs and want to see them all, you can put in 400 as the number.

A is a single character, indicating in what 'retrieval style' the items are to be shown. As explained in detail elsewhere, these are as follows:

r = record retrieval
d = data retrieval
c = caption retrieval

The default is record retrieval, so if this field is left blank the query will be in record retrieval mode.
B is a structured or Boolean query, make up of the following:

'term' and/or/minus 'term'

The terms must be in either single quotes or slashes, thus 'house' or /house/. You can use and/or/minus or the signs & | - in their place.

The query is read from left to right, thus it will assume that 'sheep' or 'goat' and 'Greece', means find all items which have either sheep and Greece or goat and Greece. If the structure starts to get complicated, then brackets may be used to indicate

the structure as with normal Boolean queries.

C is where you can specify various parameters (as described in detail elsewhere under 'The structure of A-records', the contents of an "s" string.). For present purposes, the only thing which you are likely to need is:

s = stem with the Muscat stemming algorithms

D is a free text query, which like all such queries can be in upper or lower case etc.

Thus :

A can be blank (r is assumed)

B can be blank - pure free text query

C can be blank (no stemming)

D can be blank - pure structured query

Some examples:

{\Q.200\">c>'M=sketch'"} - will present a caption list of the first two hundred sketches (M stands for 'medium').

{\Q.500\">c>'P=Mills/ JP' or 'P=Mills' and 'M=photographs'"} - will present a caption list of the first five hundred photographs taken by J.P.Mills or Mills.

{\Q.1O\">d>>s>girls combing hair"} - will present the first ten items concerning girls combing hair, in data retrieval mode, having 'stemmed' the words.

{\Q.1O\">r>'D=1936' and 'M=photographs'>s>girls combing hair"} - will present the first ten items, in record retrieval mode, concerning girls combing hair in photographs in 1936.

It should be noted that you can put ordinary terms into a structured query with this system. For instance, while in the free text system you can only search in a probabilistic way, here it would be possible to do queries of the form, "show me all the red and green cloths, but not those which also had blue in". But it is important to remember that you will need to know what the "stemmed" or suffix-stripped version of the query term is, for this is what is held in the index. For instance, it is no good looking for 'marriage', you have to use the shortened version 'marri'.

If you want to find out these forms, the best way is to use the 'q' system (as explained in Appendix F), and put in a word you want to use, and see whether it is abbreviated. If these terms are used in the structured part of the embedded query they will have to be within single inverted commas.

(Normally, if a term is not found, it is ignored. It is possible to 'force' the query to report on terms that are not found. This is done as another option when entering the system. For instance: "c-dbsys with f", will produce these reports)

If you have been taken to a list of items through such an embedded query, you can still 'mark' and save relevant items in the normal way.

APPENDIX O.

DISCATEL; THE ELEMENTARY DISC CATALOGUING SYSTEM

INTRODUCTION

The 'Discatel' system is a free-standing and simplified system that allows you to set up your own database of any size, create the record structure you would like, and define how the records are to be printed out. For those who have relatively simple data, consisting of records and fields, it is ideal for setting up a structured database.

Any of the small programs described below can be changed by using your word-processor or editor. (Use the word-processor in non-document or unformatted mode, if possible, since the formatting codes in a word-processor may cause a problem.)

SETTING UP YOUR OWN FORMAT

The discatel system sets up a simple format consisting of the 26 letter codes, A to Z. These are treated in the simple version in the ways described below.

The documents are printed on the screen and on paper in order of fields, with certain words to indicate what the fields mean etc.

Obviously, for other applications, it will be necessary to modify this simple version. Here is how to do it.

Changing the format.

The formatting specification is set up by a macro (a small program) which is in the following directory: \muscat\macros\disca-

tel and is called makef.txt

(Note that the words in square brackets in the following short programs or 'macros' are comments, not part of the program. In the actual macros they are prefaced by a backwards slash, to show that they are comments.)

\muscat\macros\discatel\makef.txt

```
[]
makef !F from
rec (      [put here the list of fields which are declared
            below. But not ids and id1, which are subfields
            of id. The order in this bracketed list is
            unimportant.]

      id
      head text
      a b c d e f g h i j k l m n o p q r s t u v w x y z
    )
id ( ids id1 )  ids = s  id1 = i        [identity number]
head=s            [general heading for the record]
text=s            [general field for text]

[data fields: alter or extend as necessary]

a=s b=s c=s d=s e=s f=s g=s h=s i=s j=s k=s l=s m=s
n=s o=s p=s q=s r=s s=s t=s u=s v=s w=s x=s y=s z=s

[remember that the order of declaration a=s to z=s
determines the order in which DISCAT will display
the records on the screen. Also remember that you
can only add new fields at the end of this list
unless you are prepared to throw the DB file away
and recreate it. If you do add further fields, 'discatel'
will not work with 'muscatel' unless the latter is
modified to include the new fields.]
!
-----------------------------------------------------
```

(The '=s' indicates that the field will be treated as a 'string' of letters or numbers, in other words not an integer or number on which calculations can be made. It is possible to declare the field to be an integer (see main manuals).

The number of fields can thus be expanded with two or more letter codes, up to a total number of 255.

How the material is to be indexed

It is necessary to specify how each field is to be indexed, and this means modifying two macros. The first is in the delbase directory, and called ispec.txt. It is as follows.

\delbase\ispec.txt

```
i *head s m1 r;     [*head is used as a 'work' field here. Change
                      with caution.]

                    [alter the following as necessary:]

i *a p 'A=';         [the prefix is A=. Used in a Boolean query
mode                     usually. Options L, U and R also useful.
                    L - put term into lower case; U means upper,
                    R means that the term is offered to the user
                    in the query expansion process.
                    See Muscat Manual 3rd ed., p. 130.]
i *b p 'B=';
i *c p 'C=';
i *d p 'D=';
i *e p 'E=';
i *f p 'F=';
i *g p 'G=';
i *h p 'H=';
i *i p 'I=';
i *j p 'J=';
i *k s r m1;      [s is suffix strip; m1 means no one letter
words]
i *l p 'L=';
i *m p 'M=';
i *n p 'N=';
i *o p 'O=';
i *p p 'P=';
i *q p 'Q=';
i *r p 'R=';
i *s p 'S=';
i *t p 'T=' u;       [ make this one upper case]
i *u s r m1;   [s is suffix strip; m1 means no one letter words]
i *v p 'V=';
i *w p 'W=';
i *x p 'X=';
i *y p 'Y=';
i *z p 'Z=';

              [put your stop list here:]

k 'about' 'after' 'again' 'against' 'all' 'an' 'and' 'ani'
'down' 'dur' 'each' 'except' 'few' 'first' 'for' 'from' 'into'
'is' 'it' 'more' 'most' 'out' 'over' 'own' 'per' 'same' 'so'
'some' 'to' 'togeth' 'too' 'under' 'until' 'up' 'us' 'veri'
```

```
'was'
'ar' 'as' 'at' 'be' 'been' 'befor' 'below' 'between' 'both'
'but' 'by' 'can' 'further' 'had' 'ha' 'have' 'how' 'if' 'in'
'no' 'nor' 'not' 'of' 'off' 'on' 'onc' 'onli' 'onto' 'or'
'other' 'such'
'than' 'that' 'the' 'their' 'then' 'there' 'through' 'which'
'while' 'why' 'will' 'with'

h *a [header code for terms = *a (use the first available data
field)]

---------------------------------
```

This sets up the ways in which fields are to be indexed. All but three are indexed for boolean/structured searching. The fields *k, *u are indexed for free text searching, being 'suffix stripped'. The field t is put into upper case.

You can modify this indexing specification in three main ways. One of these is to add further fields. Thus, having added the field 'lo' to the makef.txt macro earlier, this could be added to the above in the appropriate place, which might be after the last line (i *z p 'z=';) or, if you are starting from scratch, wherever the new field(s) have been put in the makef.txt macro.

Secondly, you can alter the way in which the fields are treated. If you want to have further fields for 'free text' searching, they can be modified to look like the *k and *u fields above.

Thirdly, the 'stop list' is a list of words which will not be indexed, because they are too common/uninteresting. You can add to or delete words from this list.

The second indexing macro to change is in the delbase directory, and called iexp.txt.

\delbase\iexp.txt

```
h goto *id \ make sure it has an id-number
write 1 atlev 1

re ( try s switch (

    [Put the different fields into the categories (case ...)
     below, depending on the retrieval needs. If there are no
fields  for a particular category, delete everything down     to
the  next case expression.]

   [Fields to be indexed as a single term (structured/Boolean
mode)]
```

```
 case *e case *l case *t case *z
    (all entry)


  [Fields to be indexed word at a time (free text mode):]

case *u case *k
    repeat(bef uc or lc l re uc or lc r entry)


   [Fields to be indexed in both of the above ways:]

case *c
    (all entry
     b repeat(bef uc or lc l re uc or lc r
              to x as *head $x s (all entry))
    )


     [Fields to be indexed as year-dates (four consecutive
digits)]

case *d
    (bef (l times 4 digit r) entry)

                  ) next )
```

This indicates that the different fields are to be treated in four different ways:
*e *l *t *z - are to be indexed as a single term (in other words, if there are more than one word in one of these fields, it should still only be indexed as one term, thus 'north America' will be treated as all one term, and not broken into 'north' and 'America')

*u *k - will be indexed a word at a time (for free text searching)

*c - will be indexed in both of the above ways

*d - will be indexed as year-dates (four digits)

If you wanted to add a some more fields, for instance the fields *a *b and *lo, then you would need to decide whether you wanted them treated in one of these special ways. If so, it would need to be added in as appropriate as 'case *a' 'case *b' 'case *lo' in the appropriate lines above.

Setting up the structured query screen

When you go into the structured query screen, and make choices, you will see the choices appear on the right side of the screen, with a prefixed letter, for example 'C:'. You may want to make this prefix more informative, putting in your own word in place of a letter. This can be done as follows:

You will need to edit the following macro:

\delbase\dspec.txt

```
h *c        [this defines the caption field]
e {
         [alter as necessary. The f 'A:' means that A: appears
          on the right hand side of the structured query system
to
         indicate terms in category *a.]
b  1 p 'A=' f 'A:';
b  2 p 'B=' f 'B:';
b  3 p 'C=' f 'C:';
b  4 p 'D=' f 'D:';
b  5 p 'E=' f 'E:';
b  6 p 'F=' f 'F:';
b  7 p 'G=' f 'G:';
b  8 p 'H=' f 'H:';
b  9 p 'I=' f 'I:';
b 10 p 'J=' f 'J:';
b 11 p 'K=' f 'K:';
b 12 p 'L=' f 'L:';
b 13 p 'M=' f 'M:';
b 14 p 'N=' f 'N:';
b 15 p 'O=' f 'O:';
b 16 p 'P=' f 'P:';
b 17 p 'Q=' f 'Q:';
b 18 p 'R=' f 'R:';
b 19 p 'S=' f 'S:';
b 20 p 'T=' f 'T:'; b 21 p 'U=' f 'U:';
b 22 p 'V=' f 'V:';
b 23 p 'W=' f 'W:';
b 24 p 'X=' f 'X:';
b 25 p 'Y=' f 'Y:';
b 26 p 'Z=' f 'Z:';
/
--------------------------------
```

If field 'T' was going to be a field for the time of year, you could change that line to:

```
b 20 p 'T=' f 'Time:';
```

and then this would appear on the right hand side of the screen, when one had chosen a term in that field, for instance 'Time: morning'.

Printing out the results.

You can change the way in which a record is printed out on the screen. This is specified by the macro called pspec.txt in the delbase directory, as follows:

\delbase\pspec.txt

```
d d (a) g1;
d *rec (t) g0;
d *id (a) k;
d *ids (a);
d *id1 (f) '.' (il) z '-';
d *head (a) g1 '\v' + '\n';
d *text (a) g1;

    [put any other d-directives here, for example, as follows:]

d *c (a) 'Title: \v' l=12 + '\n' l=0; [highlight the caption]
d *d (f) g1 'Date:       ' l=12 (il) ', ' (t) l=0;
d *e (f) g1 'Region:     ' l=12 (il) ', ' (t) l=0;
d *l (f) g1 'Country:    ' l=12 (il) ', ' (t) l=0;
d *q (a) g1 'Duration:   ' l=12 + l=0;
d *t (f) g1 'Series:     ' l=12 (il) ', ' (t) l=0;
d *u (a) g1 u12 1         l=12 + l=0;
d *z (f) g1 'Tape no.    ' l=12 (il) ', ' (t) l=0;
d *k (f) g1 'Topics:     ' l=12 (il) ', ' (t) l=0;
-----------------------------------------------------------
```

[In brief, the various letters and numbers above mean the following. For a fuller description, see the section on printing within Muscatel, in the **Muscat Manual** , 3rd edn., pp.189-193.]

```
   g1  - put out on a line by itself with 1 blank line preceding
   '....' - output text
   '\v'...'\n' - highlight the material between
   l=12   - if this line overflows indent overflow material by
            12 spaces (similarly l=0).
   +  - output the actual field
    (f)   - what to do if the field is the FIRST of a list
of fields with the same code
   (l)  - similarly LAST
```

```
   (i)  - similarly INTERMEDIATE
   (il) - means (i) or (l)
   (t)  - how a terminate a list of fields with the same code
   (a)  - means (f) or (i) or (l)
   u12 1 - put the next bit of data at character
           position 12 (counting from 0). For example...
```

The small test database 'sample.txt' shows how the above looks; it can be modified to fit your data, since obviously something developed for a video tape library, which is what the example format above was developed for, will not be appropriate.

Another macro specifies how the record is to be printed out on paper. It is called print.txt, in the \muscat\macros\discatel directory, as follows:

\muscat\macros\discatel\print.txt

```
<from/a/r,to/k,width/k,opts/k>
copy <d->pspec to !pspec from2
w<width$70>
<opts>
!
print <from> to <to$*> with !pspec
```

This basically takes the screen print specification and prints it out, so it is unlikely that you will need to modify it.

HOW TO ENTER DISCATEL AND SET UP YOUR OWN DATABASE

Once you have a formatting system, or using the preliminary one that is available, you can enter the system, as follows:

Go into the 'Delbase' directory.

Then type muscat discatel

You will now be in the discatel system, with a 'muscat>' prompt.

(If you want to work in another directory of your own, you can do so, but will need to type c-system when you get the muscat> prompt, otherwise the input is assumed to be that from the current directory.)

In order to set up a database, you type

c-create

You will then get a prompt, asking for the size in bytes. If you want a database of 100k , then type 100000.

If at a later point you want to extend this database to add more material, this can be simply done by typing
c-extend

When you will again be asked to specify the number of bytes that you want to add. (In other words, if you already have a database of 100000 and want to double its size to 200000, you would c-extend by another 100000.)

Once you have an empty database set up, then get your text file which must have the suffix '.txt,, for instance 'sample.txt'.

Build the text file with

c-build (e.g. c-build sample to sample)

Number it with c-number (e.g. c-number sample to sample1)

(to start with you can answer 'R' and 100 to the queries about record number and where to start from. If you add further records, remember that their numbers must not overlap with earlier ones, otherwise the earlier ones will be overwritten.)

Then add the file into your database with c-add

(for example, c-add sample1).

This is an updatable databse, so if you want to add a further file, you can just do the same as above, adding in another file with c-add.

Once you have a small database, you can enter it by going:

c-disc

When you want to leave it, select the 'Exit' box or type the letter x.

Then to leave discatel, type 'stop'.

You can examine and modify individual records within the database, as explained in the manual.

CHANGING THE INTRODUCTORY SCREEN PAGES

When discatel sets up the database, it automatically adds the

introductory pages which are contained in intro.txt. These are as follows:

\delbase\intro.txt

```
[| A.1 - adds extra text
   A.2 - take each line of the form:

   {g0}A: { :1">enter text for A>lc"}

and alter as necessary. 'A:' is what appears on the screen,
'enter text for A' is what appears on the top line. 'lc' means
treat what is typed by the user as lower case, with an initial
capital.]

*id A.1
*head This is the Discatel retrieval system

*text
{g0} \vF\nree text query:                {u 34 1 \A.0\=f"sl"}
{g0} \vS\ntructured query:               {u 34 1 \A.2\=s}
{g0} \vB\noth free text with structured: {u 34 1 \A.2\=b"sl"}

#

*id A.2
*head Structured Query:

*text

Select one or more of the following categories:
{g0}
{g0}A: { :1">enter text for A>lc"}
{g0}B: { :2">enter text for B>lc"}
{g0}C: { :3">enter text for C>lc"}
{g0}D: { :4">enter text for D>lc"}
{g0}E: { :5">enter text for E>lc"}
{g0}F: { :6">enter text for F>lc"}
{g0}G: { :7">enter text for G>lc"}
{g0}H: { :8">enter text for H>lc"}
{g0}I: { :9">enter text for I>lc"}
{g0}J: {:10">enter text for J>lc"}
{g0}K: {:11">enter text for K>lc"}
{g0}L: {:12">enter text for L>lc"}
{g0}M: {:13">enter text for M>lc"}
{g0}N: {:14">enter text for N>lc"}
{g0}O: {:15">enter text for O>lc"}
{g0}P: {:16">enter text for P>lc"}
{g0}Q: {:17">enter text for Q>lc"}
{g0}R: {:18">enter text for R>lc"}
```

```
{g0}S: {:19">enter text for S>lc"}
{g0}T: {:20">enter text for T>u"}
{g0}U: {:21">enter text for U>lc"}
{g0}V: {:22">enter text for V>lc"}
{g0}W: {:23">enter text for W>lc"}
{g0}X: {:24">enter text for X>lc"}
{g0}Y: {:25">enter text for Y>lc"}
{g0}Z: {:26">enter text for Z>lc"}

#
-----------------------------------------
```

You may want to modify this, adding help pages, or expanding the introductory materials. A fuller explanation of how this is done is contained in the Manual, Appendix K.

TRYING OUT THE SYSTEM

Before modifying anything, it is worth just seeing what a small sample set of data looks like when put through the supplied macros. There is a small set of 50 records in \delbase\sample.txt. These records describe videotapes in a library. The first five records are as follows:

```
*c 'The Other Kenya':
*d 9.1980
*e E. Africa
*l Kenya
*q 50 mins
*t Horizon
*u People who live on the land and in the slums of modern
Kenya ("The side the tourist doesn't see").
*z 1
#

*c 'Behind the Horoscope':
*d 11.1980
*q 50 mins
*t Horizon
*u A French psychologist's research into the facts and
fictions of astrology.
*z 1
#

*c 'Colombian Roots':
*d 10.1980
*e S. America
*l Colombia
*q 50 mins
```

```
*t The World About Us
*u Tracing the origins of Colombian traditions and society,
within the three main ethnic groups.
*z 2
#

*c 'Walkabout to Hollywood':
*d 11.1980
*e Oceania
*q 50 mins
*t The World About Us
*u An Australian Aborigine who is now an actor in Hollywood
tries to reconcile the two very different worlds in which he
lives.
*z 2
#

*c 'Roots': Origins of American folk music;
*d 12.1980
*e N. America
*q 50 mins
*t The World About Us
*u Three folk festivals visited.
*z 2
#
```

You could go into discatel, as described above, then create a small database of 100k, and build, number and add in this file, sample.txt. Then you can search for one of these records and see how it appears on the screen, and what it looks like when printed out. (To print out directly from the screen, press 'control' and function key 7 together). You will then see what the indexing and printing specifications described above actually do to a record in this format.

WHERE THE SYSTEM IS

The discatel system is contained in two different places. Most of the programs are in a directory called DELBASE.

(Note: when copied from the installation disc, the DELBASE directory is a sub-directory of the MUSCAT directory. It is not necessary that it is kept there, so you can make a directory off your root called delbase, and copy the files into that once they are on your machine. In the following, it is assumed that you have set up Delbase as a sub-directory off your root directory.)

The Delbase directory contains the following programs:

sample.txt (a small sample to try out the system)

pspec.txt (how the material is to be printed out)
intro.txt (the introductory screen pages of choices etc)
db.da (the current database, if there is one)
dspec.txt (what appears on the screen)
ispec.txt (how a document is to be indexed)
iexp.txt (how a document is to be indexed)
introduc.txt (a longer version of the introduction screen pages)

In a separate sub-directory of macros, \muscat\macros\discatel , are contained a number of the usual macros, as follows:

create.txt getrecs.txt f8.txt system.txt print.txt q.txt
listm.txt makef.txt del.txt list.txt add.txt update.txt
disc.txt f7.txt number.txt build.txt index.txt extend.txt
getmrecs.txt f1.txt init.txt

The only one of these that you may want to change is makef.txt, in other words the macro to make a format and also print.txt, the printing specification.

CONCLUSION

'Discatel' is a relatively simple system for setting up a database structure and adding records. It can be used alongside the more powerful systems described in the Manual. The Manual describes in detail some of the stages alluded to above, for instance building, numbering and adding records.

'Discatel' may be used in conjunction with 'elementary muscat' or 'Muscatel' to sort, multiply, print records, as described in Appendix P below. An interactive, screen-driven, version, is available in the CDSi system, described in Appendix V below.

APPENDIX P.
USING DISCATEL WITH MUSCATEL

Introduction

One of the considerable advantages of 'discatel' is that, as long as you confine yourself to the 26 single letter codes, *a to *z, then all the pre-written programs in 'muscatel' (elementary muscat), can be used without any problems. A detailed description of Muscatel is contained in the Muscat Manual, p.207. Here we will give one or two examples of how this would be done.

SORTING AND PRINTING FILES

Let us suppose that you were working on a library of video tapes, in which the various fields were as follows:

```
*t    name of the series
*c    title of the program
*u    content and short description
*k    keywords and categories
*l    locality (country)
*e    ethnographic area  (e.g. E.Africa)
*q    length of the program
*z    tape reference number
*d    date of making of film
```

Suppose that you had set up a format and indexing system to fit these categories (as described in the 'Discatel' system). You then type in the following three records into a text file (which we can call 'video.txt' for this exercise):

```
*c 'The Other Kenya':
*d 9.1980
*e E. Africa
*l Kenya
*q 50 mins
*t Horizon
*u People who live on the land and in the slums of modern
Kenya ("The side the tourist doesn't see").
*z 1
*k tourism
#


*c 'Behind the Horoscope':
*d 11.1980
*q 50 mins
*t Horizon
*l France
*u A French psychologist's research into the facts and
fictions of astrology.
*z 1
*k magic
#

*c 'Colombian Roots':
*d 10.1980
*e S. America *l Colombia
*q 50 mins
*t The World About Us
*u Tracing the origins of Colombian traditions and society,
```

within the three main ethnic groups.
*z 2
*k ethnicity *k music
#

You now want to print these out in various different ways, which involves firstly re-sorting them, and then specifying an appropriate way they should appear in a print-out.

A complete list of titles, alphabetically

Firstly, you might want a print-out of all the videos in alphabetical order of title of the series. In other words you want to sort on the *c fields. In the print-out, you might decide that it would look neater if the miscellaneous notes (*n), the persons named in the record (*p) and the ethnographic area (*e) were left out.

To do this, enter delbase, where your file is, and enter muscatel by typing

muscat el

Then type:

g-build video to video

This builds the file, which can then be sorted by typing:

g-sort video to video1 fields *c kill *n*p*e

This will sort the file on the *c field and delete the *n *p *e fields from the records.

You now need to print this out, in a way that would be suitable.

You could devise a print specification, calling it pspec1.txt, and keeping it in the Delbase file.

It might look as follows:

```
w 65                (sets the width of the record)
o *rec *c *t *u *k *l *q *z       (gives the order of printing)
d L1   (a)  l=0 g1;
d L0   (a)  l=8 g0;
d *rec (t) p.;
d *c *t *u *k *l *z *q *d (a) s + ',';
d *z (a) ' Tape ' (t) ',' ;
```

(the above are some print directives, as explained in the Disca-

tel and Muscatel documentation elsewhere, which specify the placing of the fields on the page etc.)

Now that you have a print specification, called pspec1.txt, you can print out the records which you have sorted as follows:

g-print video1 to video2 with pspec1

If you now leave muscatel, by going 'stop', and look at the file video2.txt with your word-processor, you will see the following:

'Behind the Horoscope':.
Horizon, A French psychologist's research into the facts and fictions of astrology., magic, France, 50 mins, Tape 1, 11.1980.

'Colombian Roots':.
The World About Us, Tracing the origins of Colombian traditions and society, within the three main ethnic groups., ethnicity, music, Colombia, 50 mins, Tape 2, 10.1980.

'The Other Kenya':.
Horizon, People who live on the land and in the slums of modern Kenya ("The side the tourist doesn't see"., tourism, Kenya, 50 mins, Tape 1, 9.1980.

You can then edit this if necessary, and print it out in the normal way.

A complete list of titles classified by location.

Or you might decide you wanted the list sorted by location rather than title. To do this you would take the same file video.txt and do the following:

You would enter delbase, where your file is, and enter muscatel in the same way by typing:

muscat el

Then type:

g-build video to video

This again builds the file, which can be sorted by typing:

g-sort video to video1

If, as often happens, there are several different references to a field within one record, in other words repeated fields, it may be useful to replace one record containing such repeated fields with several records.

In this case, for example, there might be more than one *e field referred to in a film record. To do this, you would precede the sorting stage above by using the 'multiplying records' command, g-mult0 and g-mult1, as described in the Muscat Manual,p.179. In this instance, therefore, we would type the following:

```
g-mult0 video to video1 field *e
```

Thus, if a record contained more than *e a duplicate would be made, so that it could be indexed in several places.

Then you would proceed by typing:

```
g-sort video1 to video2 fields *e*l kill *u*t*d*q*p*n
```

This will sort on ethnographic area (*e) first, and then within that on locality (*l). It will delete the fields after "kill", which are not needed for this particular print-out.

You would then print out the result by typing:

```
g-print video2 to video3 with pspec2
```

The print specification, which you can set up by typing as pspec2.txt in the same directory, could, for instance, be the following.

```
w 65
o *rec *c*t*u*k*p*e*l*q*n*z
d L2 (a) l=0 g1;
d L2 (a) l=4 g0;
d L0 (a) l=8 g0;
d *rec (t) p.;
d *c*t*u*k*p*e*l*z*q*n*d (a) s + ',' ;
d *z (a) 'Tape ' (t)  ',';
```

If you followed the stages above, on the small sample of three records above, the result would look as follows:

```
    E. Africa. Kenya.
        'The Other Kenya':, tourism,Tape 1.
    S. America. Colombia.
        'Colombian Roots':, ethnicity, music,Tape 2.
```

A list of titles classified by topic

Finally, you might like to have the titles arranged by topic, that is by the *k field. Using the same print specification as the previous example, but "killing" different fields, you could type the following series of commands:

```
muscat el

g-build video to video

g-mult0 video to video1 field *k

g-sort video1 to video2 fields *k*c kill *t*d*q*p*c*n

g-print video2 to video4 with pspec2
```

(In the last line, we have used 'video4' so that the final result does not overwrite the video3.txt file from the previous exercise, which you might want to keep.)
The result of doing the above on the sample three records would look as follows:

```
    ethnicity. 'Colombian Roots':.
        Tracing the origins of Colombian traditions and society,
        within the three main ethnic groups., S. America,
        Colombia,Tape 2.
    magic. 'Behind the Horoscope':.
        A French psychologist's research into the facts and
        fictions of astrology., France,Tape 1.
    music. 'Colombian Roots':.
        Tracing the origins of Colombian traditions and society,
        within the three main ethnic groups., S. America,
        Colombia,Tape 2.
    tourism. 'The Other Kenya':.
        People who live on the land and in the slums of modern
        Kenya ("The side the tourist doesn't see")., E. Africa,
        Kenya,Tape 1.
```

Again, this could be edited with a word-processor if necessary.

Conclusion

There are a number of other useful and relatively simple programs described in the Muscat Manual for listing out, merging, numbering, retrieving and other tasks. By using 'Discatel' it is possible to use all of these ready made macros.

APPENDIX Q.

HOW TO ADD NEW FIELDS.

Introduction

It is possible to modify the format and indexing system so that new fields are added. A simple way to do this automatically, is described in Appendix V. This only works with the simpler, 26-field, indexing structure. It is therefore worth describing what needs to be done to change a more complex structure, if you should need that.

Several changes are needed. The format has to be changed; the indexing macro has to be modified; the print specifications have to be altered and the 'build' macro has to be modified. The way in which each of the fields is treated is outlined in the Manual and in Appendix L. Here we will give an example to show how a change can be made.

We decided that a different set of data needed a different field structure. Instead of producer, collector and acquirer of museum objects, we wanted to devise a structure which would deal with a census of humans and their livestock and crops. So we needed the fields 'person' , 'animal' and 'crop'.

Within these fields we wanted a number of sub-fields. So we chose some letters which had not been used before, a b h j o q. (It would of course be possible to use a combination of other letters, ab abc abcde etc.) We wanted each of these to be a 'string' field, and each of them to be indexed in the same way as the *u and *k fields, that is word by word. The way we modified the format can be seen as follows.

Changing the format

The original formatting macro, \muscat\macros\cds\makef.txt was as follows:

```
[ ]
makef $formats\cds from
rec ( num i c store u m t k kp ke kl kd
      z prod coll acq g qv ns recr g0 g1 )
num = i    \ Record identity generated by Muscat
i ( is i1 i2 )  is = s  i1 = i i2 = i  \ identity number
c = s      \ general class of object
store ( store1 stored )  \ storage location
store1 = s  stored = s   \ main part and detail
u = s      \ simple name
m = s      \ medium (e.g. photograph)
```

```
t = s     \ full name or description
k = s     \ keywords
kp = s    \ person or people keywords
ke = s    \ ethnographic group keywords
kl = s    \ locality keywords
kd = s    \ date keywords
z = s     \ size (ignore if photo contains a ruler)
prod ( f p d e l r n )  \ production
coll ( f p d e l r n )  \ collection
acq  ( f p d e l r n co )  \ acquisition
g ( i store u m t k kp ke kl z n )  \ info about parts
pt = s  \ title of part
qv = s    \ "see also"
ns = s    \ general notes
f ( f1 fd )  \ "function" word
f1 = s  fd = s  \ main part & detail
p ( p1 p2 pd )  \ person
p1 = s  p2 = s  pd = s  \ surname, forename & detail
d ( d1 dd )     \ date
d1 = s  dd = s  \ main part and detail
e ( e1 ed )     \ ethnographic group
e1 = s  ed = s  \ main part & detail
l ( l1 ld )     \ locality
l1 = s  ld = s  \ main part & detail
r = s         \ reference number
n = s         \ note
recr ( rp rd )  \ recorder
rp = s  rd = s  \ person & date

g0 = s g1 = s \ general text fields
!
```

This was edited, using a word processor, to look as follows:
(the changes are underlined and in bold).

```
 [ ]
makef $formats\cds from
rec ( num i c store u m t k kp ke kl kd
      z pers anim crop g qv ns recr g0 g1 )
num = i   \ Record identity generated by Muscat
i ( is i1 i2 )  is = s  i1 = i i2 = i  \ identity number
c = s     \ general class of object
store ( store1 stored )  \ storage location
store1 = s  stored = s   \ main part and detail
a = s
b = s
h = s
j = s
o = s
q = s
```

```
u = s      \ simple name
m = s      \ medium (e.g. photograph)
t = s      \ full name or description
k = s      \ keywords
kp = s     \ person or people keywords
ke = s     \ ethnographic group keywords
kl = s     \ locality keywords
kd = s     \ date keywords
z = s      \ size (ignore if photo contains a ruler)
pers (a b h j o q p )
anim (a b h j o q p )
crop (a b h j o q p )
g ( i store u m t k kp ke kl z n )  \ info about parts
pt = s  \ title of part
qv = s     \ "see also"
ns = s     \ general notes
f ( f1 fd )  \ "function" word
f1 = s  fd = s  \ main part & detail
p ( p1 p2 pd )  \ person
p1 = s  p2 = s  pd = s  \ surname, forename & detail
d ( d1 dd )      \ date
d1 = s  dd = s  \ main part and detail
e ( e1 ed )      \ ethnographic group
e1 = s  ed = s  \ main part & detail
l ( l1 ld )      \ locality
l1 = s  ld = s  \ main part & detail
r = s          \ reference number
n = s          \ note
recr ( rp rd )  \ recorder
rp = s  rd = s  \ person & date

g0 = s g1 = s \ general text fields
!
```

This new format then has to be installed. This is done by going into Muscat and then type:

```
c-makef
```

then type

```
format \muscat\formats\cds
```

If there are errors, they will be specified, otherwise the new format will be installed.

Indexing

Next, the way in which the indexing program works will need to be modified. The original indexing macro is \muscat\macros\cds\index.txt and is as follows:

```
<from/a/r,to/k/a/r>
copy to !ispec from

i d      s v '' j '\^' m1 l ;
i *u*k   r s v '' j '\^' m1 l ;
i *m     v '' j '\^'  p 'M=' ;
i *p2    v '/ ' j '\^'  p 'S=' ;
i *pd    v '/ ' j '\^'  p 'P=' ;
i *ed    v '/ ' j '\^'  p 'E=' ;
i *ld    v ' ' j '\^'  p 'L=' ;
i *kd    p 'D=' ;
i *store1 v '=' x = - p 'V=';

h *k  \ header code for terms = *k
      \ now comes the stoplist:
k 'about' 'after' 'again' 'against' 'all' 'an' 'and' 'ani'
'down' 'dur' 'each' 'except' 'few' 'first' 'for' 'from' 'into'
'is' 'it' 'more' 'most' 'out' 'over' 'own' 'per' 'same' 'so'
'some'
'to' 'togeth' 'too' 'under' 'until' 'up' 'us' 'veri' 'was'
'ar' 'as' 'at' 'be' 'been' 'befor' 'below' 'between' 'both'
'but' 'by'
'can' 'further' 'had' 'ha' 'have' 'how' 'if' 'in' 'no' 'nor'
'not'
'of' 'off' 'on' 'onc' 'onli' 'onto' 'or' 'other' 'such'
'than' 'that' 'the' 'their' 'then' 'there' 'through' 'which'
'while' 'why' 'will' 'with'
'village'
!
indexx from <from> to <to> with !ispec exp

write 1 atlev 1

not find *is s eq 'T' \ avoid indexing purely text records
re ( try switch ( case *prod $ g setto *prod()
                  case *coll $ g setto *coll()
                  case *acq  $ g setto *acq()
                  case *store1
                    s re ( aft uc bef digit l re digit re ' '
                           try ('=' re ' ' re digit) r to x
                           $ x s ( bef digit
                                     re hold (not times 5 digit
ins '0')
                                   all entry ) )
                  case *u case *k case *c
                      s re ( bef uc or lc l re uc or lc r entry
n )
                  case *m s ( all entry )
                  case *p1
```

```
                    ( $g code *acq                          s ( all
to x as *p2 )
                      $ x s ( all entry ) ) or
                    ( s (
                              re ( bef uc or lc l re uc or lc r
entry n )
                        all to x as *pd
                      )
                      try (next code *p2 to y $x s (e ins '/ '
ins y))
                      $x s ( all entry )
                    )
                  case *kp
                     s ( re ( bef uc or lc l re uc or lc r entry
n )
                      all
                      to x as *pd $x s ( all entry )
                    )
                  case *e1
                  case *ke
                    s ( code *ke or $ g not code *acq
                         re ( bef uc or lc l re uc or lc r entry
n )
                      all
                      to x as *ed $x s ( all entry )
                    )
                  case *l1
                  case *kl
                    s ( code *kl or $ g not code *acq
                         re ( bef uc or lc l re uc or lc r entry
n )
                      b
                      try ( bef '(' l e r del )
                      all to x as *ld $x s ( all entry )
                    )
                  case *d1
                  case *kd
                    s ( code *kd or $ g not code *acq
                      all to x as *kd
                      $ x s (
                              try ( h bef '-'
                                    ( bef('.'l bef '-'r to bit
                                         bef substring bit)
                                      all exch bit )
                                    or ( all exch '' ) )
                                 re ( b l bef '.' r to bit n r
del
                                    (bef '/') or e ins '/' ins
bit )
                              b l times 4 digit r entry
```

```
                times 2 ( '/' (digit digit) or (ins '0' digit) r
entry )
                                    )
                              )
                    )
      next
   )
!
```

This was modified as follows (modifications in bold, underlined):

```
<from/a/r,to/k/a/r>
copy to !ispec from

i d      s v '' j '\^' m1 l ;
i *u*k**a*b*h*j*o*q*p**   r s v '' j '\^' m1 l ;
i *m     v '' j '\^'  p 'M=' ;
i *p2    v '/ ' j '\^'  p 'S=' ;
i *pd    v '/ ' j '\^'  p 'P=' ;
i *ed    v '/ ' j '\^'  p 'E=' ;
i *ld    v ' ' j '\^'  p 'L=' ;
i *kd    p 'D=' ;
i *store1 v '=' x = - p 'V=';

h *k  \ header code for terms = *k
      \ now comes the stoplist:
k 'about' 'after' 'again' 'against' 'all' 'an' 'and' 'ani'
'down' 'dur' 'each' 'except' 'few' 'first' 'for' 'from' 'into'
'is' 'it' 'more' 'most' 'out' 'over' 'own' 'per' 'same' 'so'
'some'
'to' 'togeth' 'too' 'under' 'until' 'up' 'us' 'veri' 'was'
'ar' 'as' 'at' 'be' 'been' 'befor' 'below' 'between' 'both'
'but' 'by'
'can' 'further' 'had' 'ha' 'have' 'how' 'if' 'in' 'no' 'nor'
'not'
'of' 'off' 'on' 'onc' 'onli' 'onto' 'or' 'other' 'such'
'than' 'that' 'the' 'their' 'then' 'there' 'through' 'which'
'while' 'why' 'will' 'with'
'village'
!
indexx from <from> to <to> with !ispec exp

write 1 atlev 1

not find *is s eq 'T' \ avoid indexing purely text records
re ( try switch ( case **pers** $ g setto **pers**()
                  case **anim** $ g setto **anim**()
                  case **crop**  $ g setto **crop**()
                  case *store1
                    s re ( aft uc bef digit l re digit re ' '
```

```
                          try ('=' re ' ' re digit) r to x
                          $ x s ( bef digit
                                    re hold (not times 5 digit
ins '0')
                                  all entry ) )
                    case *u case *k case *c case *a case *b case
*h
                    case *j case *o case *q
                        s re ( bef uc or lc l re uc or lc r entry
n )
                  case *m s ( all entry )
                  case *p1
                    ( $g code *crop
                      s ( all to x as *p2 )
                      $ x s ( all entry ) ) or
                    ( s (
                             re ( bef uc or lc l re uc or lc r
entry n )
                          all to x as *pd
                        )                          try (next code
*p2 to y $x s (e ins '/ ' ins y))
                        $x s ( all entry )
                    )
                  case *kp
                     s ( re ( bef uc or lc l re uc or lc r entry
n )
                        all
                        to x as *pd $x s ( all entry )
                      )
                  case *e1
                  case *ke
                    s ( code *ke or $ g not code *crop
                         re ( bef uc or lc l re uc or lc r entry
n )
                        all
                        to x as *ed $x s ( all entry )
                      )
                  case *l1
                  case *kl
                    s ( code *kl or $ g not code *crop
                         re ( bef uc or lc l re uc or lc r entry
n )
                        b
                        try ( bef '(' l e r del )
                        all to x as *ld $x s ( all entry )
                      )
                  case *d1
                  case *kd
                    s ( code *kd or $ g not code *crop
                        all to x as *kd
```

```
                        $ x s (
                                try ( h bef '-'
                                      ( bef('.'l bef '-'r to bit
                                           bef substring bit)
                                        all exch bit )
                                      or ( all exch '' ) )
                                   re ( b l bef '.' r to bit n r
del
                                      (bef '/') or e ins '/' ins
bit )
                                  b l times 4 digit r entry
                 times 2 ( '/' (digit digit) or (ins '0' digit) r
entry )
                                )
                           )
                    )
      next
    )
!
```

Printing on the screen

The way in which the records will appear on the screen is controlled by the macro \muscat\cds\pspec.txt. This is as follows:

```
\   w70 \   j\ j^
d *rec (t) g0;
\   d *num (a) 'Item ' + ':' g1;

d *i1 (a) ':'; d *i2 (a) z '-';
d *store (a);
d *c (a) + ':' s;
d *u (a) s '\v' l+2 + p. l-2 '\n';
d *m (a) s + p.;
d *t (a) l+4 g1 + p. l-4 g0;
d *prod (a) g0 l+14 'Production:' u14 1 + p. l-14;
d *coll (a) g0 l+14 'Collection:' u14 1 + p. l-14;
d *acq (a) g0  l+14 'Acquisition:'u14 1 + p. l-14;
d *k (f) g0 (il) ', ' (t) '.';
d *kp *ke *kl *kd (f) s (il) ', ' (t) '.';
d *z (a) ' Size: ' ;
d *l*p*d*e*f (f) s (il) ', ' (t) p; ;
d *ld*pd*dd*ed*fd (a) s '(' + ')';
d *p (a) [*p1*pd (a) k][*p2 (a)]
         s +
         [*p2 (a) k][*p1 (a)][*pd (a) s '(' + ')']
         s + p; ;
d *n (a) s '(' + ')';
```

```
d *qv*ns*r (a) g0 '(' + ')';
d *recr (a) g1 '[' + ']';
d *rd (a) ':';
d *g (f) l+8 g1 + l-8 (il) l+8 g0 + l-8;

d *g0 (a) g0; d *g1 (il) g1;
```

(Note: the above macro presents a screen with the number of the record not shown; another macro 'on.txt' is identical to the above but contains as line four:

```
d *i (a) 'Record number ' + g2;
```

This has the effect of printing the record number at the top of the screen when one goes into the database with 'numbers on'. This is useful for certain types of error correction.)

This was modified as follows (changes underlined):

```
\   w70
\   j\ j^
d *rec (t) g0;
\   d *num (a) 'Item ' + ':' g1;
d *i1 (a) ':'; d *i2 (a) z '-';
d *store (a);
d *c (a) + ':' s;
d *u (a) s '\v' l+2 + p. l-2 '\n';
d *m (a) s + p.;
d *t (a) l+4 g1 + p. l-4 g0;
d *pers (a) g0 l+14 'Person:' u14 1 + p. l-14;
d *anim (a) g0 l+14 'Animals:' u14 1 + p. l-14;
d *crop (a) g0  l+14 'Crops:'u14 1 + p. l-14;
d *k (f) g0 (il) ', ' (t) '.';
d *kp *ke *kl *kd (f) s (il) ', ' (t) '.';
d *z (a) ' Size: ' ; d *p*a*b*h*j*o*q (f) s (il) ', ' (t) p; ;
d *ld*pd*dd*ed*fd (a) s '(' + ')';
d *p (a) [*p1*pd (a) k][*p2 (a)]
         s +
         [*p2 (a) k][*p1 (a)][*pd (a) s '(' + ')']
         s + p; ;
d *n (a) s '(' + ')';
d *qv*ns*r (a) g0 '(' + ')';
d *recr (a) g1 '[' + ']';
d *rd (a) ':';
d *g (f) l+8 g1 + l-8 (il) l+8 g0 + l-8;

d *g0 (a) g0; d *g1 (il) g1;
```

The print specification for 'q' and for hard copy

Printing out on paper is controlled by the print specification \muscat\cds\dspec.txt, which is as follows:

```
w70
j\ j^
d *num (a) 'Item ' + ':' g1;
d *i1 (a) ':'; d *i2 (a) z '-';
d *i *store *c (a) s;
d *u (a) g1 + p.;
d *m (a) s + p.;
d *t (a) l+4 g1 + p. l-4 g0;
d *prod (a) g0 l+4 'Prod: ' + p. l-4;
d *coll (a) g0 l+4 'Coll: ' + p. l-4;
d *acq (a) g0 l+4 'Acq: ' + p. l-4;
d *k (f) g0 (il) ', ' (t) '.';
d *kp *ke *kl *kd (f) s (il) ', ' (t) '.';
d *z (a) ' Size: ' ;
d *l*p*d*e*f (f) s (il) ', ' (t) p; ;
d *ld*pd*dd*ed*fd (a) s '(' + ')';
d *pd (a) ', ';
d *n (a) s '(' + ')';
d *qv*ns*r (a) g0 '(' + ')';
d *recr (a) g1 '[' + ']';
d *rd (a) ':';
d *g (a) l+8 g0 + l-8;
```

This was modified as follows (changes underlined):

```
w70
j\ j^
d *num (a) 'Item ' + ':' g1;
d *i1 (a) ':'; d *i2 (a) z '-';
d *i *store *c (a) s;
d *u (a) g1 + p.;
d *m (a) s + p.;
d *t (a) l+4 g1 + p. l-4 g0;
d *pers (a) g0 l+4 'Person: ' + p. l-4;
d *anim (a) g0 l+4 'Animals: ' + p. l-4;
d *crop (a) g0 l+4 'Crops: ' + p. l-4;
d *k (f) g0 (il) ', ' (t) '.';
d *kp *ke *kl *kd (f) s (il) ', ' (t) '.'; d *z (a) ' Size: ' ;
d *p*a*b*h*j*o*q (f) s (il) ', ' (t) p; ;
d *ld*pd*dd*ed*fd (a) s '(' + ')';
d *pd (a) ', ';
d *n (a) s '(' + ')';
d *qv*ns*r (a) g0 '(' + ')';
d *recr (a) g1 '[' + ']';
```

```
d *rd (a) ':';
d *g (a) l+8 g0 + l-8;
```

Listing out the data

To list out (print) from the built records in a file, it is necessary to modify the macro \muscat\macros\cds\list.txt

The original macro was as follows:

```
[from/a/r,to/k,gap]
print from [from] to [to$*] with

w 58
e
d d (a) s x f s;

d *rec (a) g[gap$1] e (t) ' #';
d *i *store *c *num *t *k *m *z *n *ns *p *d *l *r *n
  (f) s x f s (il) ' *' f s;

d *i1 (a) ':'; d *i2 (a) z '-';
d *f1 *p1 *d1 *l1 *e1 *store1 *rp *is
  (il) s x f s;

d *fd *pd *dd *ld *ed *stored *rd
  (a) s '<';

d *p2
  (a) s '/';

d *g *prod *coll *acq *recr *ns *g0 *g1
  (a) g0 x f s;

d *f (f) e s (il) s x f s;
!
```

The modified version, with the changes underlined, is thus:

```
[from/a/r,to/k,gap]
print from [from] to [to$*] with

w 58
e
d d (a) s x f s;

d *rec (a) g[gap$1] e (t) ' #';
d *i *store *c *num *t *k *m *z *n *ns *p *d *l *r *n
  (f) s x f s (il) ' *' f s;
```

```
 d *i1 (a) ':'; d *i2 (a) z '-';
d *f1 *p1 *d1 *l1 *e1 *store1 *rp *is
  (il) s x f s;

d *fd *pd *dd *ld *ed *stored *rd
  (a) s '<';

d *p2
  (a) s '/';

d *g *pers *anim *crop *recr *ns *g0 *g1
  (a) g0 x f s;

d *f (f) e s (il) s x f s;
!
```

Building the data

Finally, it is necessary to modify the macro which specifies how the record is to be built, \muscat\macros\cds\build.txt. The original macro was as follows:

```
[from/a/r/h,to/k/a/r]
build from [from] to [to] with

m *f/p *p/p *d/p *l/f *store/r *recr/f

a *coll *f1 < *fd #  \ Here and below, '<' is a detail indicator
a *prod *f1 < *fd #
a *acq *f1 < *fd #
a *e *e1 < *ed #
a *f *f1 < *fd #
a *p *p1 / *p2 < *pd #
a *l *l1 < *ld #
a *d *d1 < *dd #
a *store *store1 < *stored #
a *recr *rp / *rd #
a *g *i1 - *i2 = *i2 #
i *i *is . *i1 : *i1 - *i2 = *i2 #
!
```

This was modified to take account of the new field structure as follows (changes underlined):

```
[from/a/r/h,to/k/a/r]
build from [from] to [to] with

m *f/p *p/p *d/p *l/f *store/r *recr/f

a *pers *f1 < *fd #  \ Here and below, '<' is a detail indicator
```

```
a *anim *f1 < *fd #
a *crop *f1 < *fd #
a *e *e1 < *ed #
a *f *f1 < *fd #
a *p *p1 / *p2 < *pd #
a *l *l1 < *ld #
a *d *d1 < *dd #
a *store *store1 < *stored # a *recr *rp / *rd #
a *g *i1 - *i2 = *i2 #
i *i *is . *i1 : *i1 - *i2 = *i2 #
!
```

Conclusion

This completes the changes that have to be made for this adjustment. It should be stressed that changes should not, if possible, be made to the formatting and indexing macros once you have already built some data into a database. It is very easy to make a small mistake and to corrupt the earlier material. Thus it is best to start afresh, creating a new database and building the data anew if a change is made to the format. If it is absolutely necessary to add in a new field after a database has been partly created, make sure that the field is added at the end of the declarations in the makef.txt macro.

APPENDIX R

FULL TEXTS OF THE CAMBRIDGE DATABASE MACROS

All these are in the \muscat\macros\cds directory, with the extension .txt. They are given here in alphabetic order. Since some lines have been moved to the left, as they were otherwise too wide to print in this Manual, you are advised to look at the originals in the computer if you need the exact spacing.

ACQR

Sorts records by archival reference (*r) in the acquisition (*acq) field.

```
<from/a/r,to/k/a/r>
keyx from <from> to $WORK\f3 exp
write 1 atlev 1
find *acq g find *r write 1 atlev 0
!
u-sort $WORK\f3 to $WORK\f4
retx $WORK\f4 to <to> exp
```

```
lev 0
!
```

ADD

Adds a file to the database.

```
<from/a/r,style/k>
c-index<style> <from> to $WORK\f1
keyx $WORK\f1 to $WORK\f2 exp
(lev 1 goto *i write 1 atlev 1) or (setf /0 write 1)
!
u-sort $WORK\f2 to $WORK\f3 opt fo
DBadd from $WORK\f1 from2 $WORK\f3 file <d->db with
p i *is !
```

BATCHADD

To batch add a file, running build/liste1/number/add

```
<from/a/r,lettercode/k/a/r,startingat/k/a/r>
c-build <from> to $work\fr
c-liste1 $work\fr to $work\fa
c-build $work\fa to $work\fs
c-number $work\fs to $work\fr lettercode
                  <lettercode> startingat <startingat>
c-add $work\fr
```

BK1

To create a raw contents list for a book.

```
<from/a/r,to/k/a/r>
echo creates raw contents list for book
keyx <from> to $WORK\f1 exp
selectto x *i *u *r
$x write 1
!
print $WORK\f1 to <to> with
d *rec (a) g0;
d *i (a) '*g0 {/' + '/}';
d *i1 (a) '.'; d *i2 (a) z '-';
d *u (a) s;
d *r (a) s '(' + ')';
!
```

BKSPLIT

Splits books and manuscripts into a text and record part

```
<from/a/r,tor/k/a/r,tot/k/a/r>
retx <from> to F0 exp
goto *is s (l e r exch 'R')
goto *prod
!
c-kill F0 to <tor> fields *g1
edx <from> <tot> exp
repeat ( read 0 to x
         $x selectto y *i*u*g0*g1*kd
         $y write 1 )
!
```

BUILD

To build a record so that it is ready to go into a database.

```
[from/a/r/h,to/k/a/r]
build from [from] to [to] with
m *f/p *p/p *d/p *l/f *store/r *recr/f

a *coll *f1 < *fd #  \ Here and below, '<' is a detail indicator
a *prod *f1 < *fd #
a *acq *f1 < *fd #
a *e *e1 < *ed #
a *f *f1 < *fd #
a *p *p1 / *p2 < *pd #
a *l *l1 < *ld #
a *d *d1 < *dd #
a *store *store1 < *stored #
a *recr *rp / *rd #
a *g *i1 - *i2 = *i2 #
i *i *is . *i1 : *i1 - *i2 = *i2 #
!
```

CHECK0

To check that certain major fields are in a record.

```
<from/a/r>
retx from <from> to $WORK\fz exp
not (
(goto *i adv not code *i) and
(goto *c adv not code *c) and
(goto *t adv not code *t) and
goto *prod
```

```
)
!
```

CHECKID

To check the identity fields in a file of records.

```
<from/a/r>
checkid from <from> to $WORK\fz with
knr
i *is
e1000
!
t from
Faulty recs (if any) left in $WORK\fz
!
```

CREATE

To create an empty database (DB), prompting for size in megabytes

```
<kilobytes/a/r>
DBcreate <d->DB bytes <kilobytes>000 blocksize 6144 with
k1
!
c-build $cds\intro to !R
c-add !R
```

DATEIND

To sort a batch of records by the *prod *d field (production date).

```
<from/a/r,to/k/a/r>
| to sort a batch of records by the *prod*d field :
keyx from <from> to $WORK\f0 exp
write 1 atlev 1
find *prod  g find *d1 s (

bef digit l bef '-' or q r to x
$z setto *d1 ''
$x s re ( bef digit l re digit r to y $z s (ins y ins '/') )
$z write 1 atlev 0

)
!
u-sort $WORK\f0 to $WORK\f1
retx $WORK\f1 to <to> exp
lev 0
!
```

DBSYS

To go into the database (DB) system.

```
<on/k,add/k,marksto/k,opts/k,with/k,numbers/k>
disc file <d-><on$db> pspec $cds\<numbers$pspec> with
s *u
h *u
e {
m69
<opts>
b 1 p 'D=' f 'Date:';
b 2 p 'P=' f 'Person:';
b 3 p 'L=' f 'Locality:';
b 4 p 'E=' f 'the';
b 5 p 'M=' f 'Recording medium:';
b 6 p 'V=' f 'Frames:';
b 7 p 'S=' f 'Source:';
v
/
<with>
-
m 1 15
d 1 15
!
```

(Note: The 'v' in the list of <opts>, changes the screen so that returning to a previous screen is achieved by pressing 'Esc', and help screens are found by pressing function key 7 (F7). If the 'v' is removed, a version which has 'R' for return, and 'H' for help, will be installed.)

(Note: Various settings can be made after the <with> line. The minus sign, - , alters the screen setting; 'v' could be inserted here to automatically link the computer to a videodisc; z could be put in to turn the setting to one appropriate for a monochrome (non-colour) screen; the letter i could be inserted, which would inhibit the system so that none of the function keys work when interrogating a database. Thus one could have a list, such as:

```
<with>
-
z
v
i
m 1 15
d 1 15
!
```

DEL

To delete a single record from a database.

```
<recnum/a/r>
c-build to $WORK\f1 from
#H *i
*i <recnum> #
!
dbadd from $WORK\f1 file <d->db with
dp
i *is
!
```

DISC

To go into the Direct Access (DA) Database.

```
<on/k,add/k,marksto/k,opts/k,with/k,numbers/k>
disc    terms    <d-><on>daterm    recs    <d-><on>darec    pspec
$cds\<numbers$pspec> with
s *u
h *u
e {
m 69
<opts>
b 1 p 'D=' f 'Date:';
b 2 p 'P=' f 'Person:';
b 3 p 'L=' f 'Locality:';
b 4 p 'E=' f 'the';
b 5 p 'M=' f 'Recording medium:';
b 6 p 'V=' f 'Frames:';
b 7 p 'S=' f 'Source:';
v
/
<with>
-
m 1 15
d 1 15
!
```

(see note at end of the DBSYS macro above, where various options and defaults which can also be set here are explained)

DUMP

A version of 'print' which shows the hierarchical structure of the fields (useful for de-bugging).

```
<from/a,to/k>
```

```
print from <from> to <to$*> with
d d (a) g0 x f s l+2 + l-2;
!
```

F1

Function key 1, plus control; saving a record to a temporary file within a database.

```
<>
c-list !rec-1 to !text-1
sed !text-1
c-build !text-1 to !rec-1
c-index !rec-1 to !work
DBadd from !work file <d->\db with
n p i *is
!
```

F6

Function key 6, plus control; to obtain a wide directory listing of the current directory.

```
<>
msdos dir /w
msdos pause
```

F5

Function key 5, plus control; to delete the current record from the database.

```
<>
DBadd from !r file <d->db with
dp i *ids
!
```

F7

Function key 7, plus control; to print out the current record.

```
<>
c-print !r to /L
```

F8

Function key 8, plus control; going out of a database temporarily into the main Muscat/Discat system.

```
<>
reenter Discat>
```

GETDISCM

To get marked records from a DA file

```
<markfile/a/r,to/k/a/r>
retx from <markfile>.mks to !temp-f1 exp
lev 1
!
fromcat !temp-f1 to !temp-numbers
getrecs to <to> from !temp-numbers recs <d->darec
delsf !temp-f1 !temp-numbers
```

GETMRECS

To get marked records from a DB file

```
<markfile/a/r,to/k/a/r>
retx from <markfile>.mks to !temp-f1 exp
lev 1
!
fromcat !temp-f1 to !temp-numbers
getrecs to <to> from !temp-numbers file <d->DB
delsf !temp-f1 !temp-numbers
```

GETRECS

To get marked records

```
<query/a/r,to/k/a/r>
copy to !temp-commands from
bfrom <query>
numbersto !temp-numbers
stop
!
q file <d->DB from !temp-commands pspec
!
getrecs file <d->DB from !temp-numbers to <to>
delsf !temp-numbers !temp-commands
```

IED1

Moving data from a group (*g) field into an identity (*i) field.

```
<from/a/r,to/k/a/r>
edx <from> <to> exp
re ( read 0 to x
     $ x($i setto *rec()
         h try ( goto *g
                 each *g ( goto *i to y as *rec*i
                           $i mate y *i/f )
               )
         mate i *i/r
         write 1)
   )
```

INDEX

Specifies the indexing that is to occur; which fields to index in what ways, and which words and fields not to index.

```
<from/a/r,to/k/a/r>
copy to !ispec from

i d      s v '' j '\^' m1 l ;
i *u*k   r s v '' j '\^' m1 l ;
i *m     v '' j '\^'  p 'M=' ;
i *p2    v '/ ' j '\^'  p 'S=' ;
i *pd    v '/ ' j '\^'  p 'P=' ;
i *ed    v '/ ' j '\^'  p 'E=' ;
i *ld    v ' ' j '\^'  p 'L=' ;
i *kd    p 'D=' ;
i *store1 v '=' x = - p 'V=';

h *k  \ header code for terms = *k
      \ now comes the stoplist:
k 'about' 'after' 'again' 'against' 'all' 'an' 'and' 'ani'
'down' 'dur' 'each' 'except' 'few' 'first' 'for' 'from' 'into'
'is' 'it' 'more' 'most' 'out' 'over' 'own' 'per' 'same' 'so'
'some'
'to' 'togeth' 'too' 'under' 'until' 'up' 'us' 'veri' 'was'
'ar' 'as' 'at' 'be' 'been' 'befor' 'below' 'between' 'both'
'but' 'by''can' 'further' 'had' 'ha' 'have' 'how' 'if' 'in'
'no' 'nor' 'not''of' 'off' 'on' 'onc' 'onli' 'onto' 'or'
'other' 'such''than' 'that' 'the' 'their' 'then' 'there'
'through' 'which' 'while' 'why' 'will' 'with' 'village'
!
indexx from <from> to <to> with !ispec exp

write 1 atlev 1

not find *is s eq 'T' \ avoid indexing purely text records
re ( try switch ( case *prod $ g setto *prod()
```

```
                  case *coll $ g setto *coll()
                  case *acq  $ g setto *acq()
                  case *store1
                    s re ( aft uc bef digit l re digit re ' '
                           try ('=' re ' ' re digit) r to x
                           $ x s ( bef digit
                          re hold (not times 5 digit ins '0')
                                   all entry ) )
                  case *u case *k case *c
                      s re ( bef uc or lc l re uc or lc r entry
n )
                  case *m s ( all entry )
                  case *p1
                    ( $g code *acq
                      s ( all to x as *p2 )
                      $ x s ( all entry ) ) or
                    ( s (
                        re ( bef uc or lc l re uc or lc r entry
n )
                          all to x as *pd
                         )               try (next code *p2 to y
$x s (e ins '/ ' ins y))
                        $x s ( all entry )
                    )
                  case *kp
                    s ( re ( bef uc or lc l re uc or lc r entry
n )
                        all
                        to x as *pd $x s ( all entry )
                      )
                  case *e1
                  case *ke
                    s ( code *ke or $ g not code *acq
                        re ( bef uc or lc l re uc or lc r entry
n )
                        all
                        to x as *ed $x s ( all entry )
                      )
                  case *l1
                  case *kl
                    s ( code *kl or $ g not code *acq
                        re ( bef uc or lc l re uc or lc r entry
n )
                        b
                        try ( bef '(' l e r del )
                        all to x as *ld $x s ( all entry )
                      )
                  case *d1
                  case *kd
                    s ( code *kd or $ g not code *acq
```

```
                        all to x as *kd
                        $ x s (
                                try ( h bef '-'
                                      ( bef('.'l bef '-'r to bit
                                            bef substring bit)
                                        all exch bit )
                                      or ( all exch '' ) )
                                   re ( b l bef '.' r to bit n r
del
                                      (bef '/') or e ins '/' ins
bit )
                                b l times 4 digit r entry
     times 2 ( '/' (digit digit) or (ins '0' digit) r entry )
                                )
                        )
                )
     next
   )
!
```

KDIND

To sort a batch of records by the date (*kd) field.

```
<from/a/r,to/k/a/r>
\ to sort a batch of records by the *kd field :
keyx from <from> to $WORK\f0 exp
write 1 atlev 1
find *kd s (

bef digit l bef '-' or q r to x $z setto *d1 ''
$x s re ( bef digit l re digit r to y $z s (ins y ins '/') )
$z write 1 atlev 0

)
!
u-sort $WORK\f0 to $WORK\f1
retx $WORK\f1 to <to> exp
lev 0
!
```

KILL

To kill (suppress) a field or fields from a file of records.

```
<from/a/r,to/k/a/r,fields/k/a/r>
```

```
keyx from <from> to <to> exp
to new
mateto new /k *rec/r <fields>
$new (next q) or write 1
!
```

KLIND

To link maps with synonyms (see klspec.txt for a description of how this is done).

```
<from/a/r,to/k/a/r>
keyx from <from> to $WORK\f0 exp
write 1 atlev 1
re(goto *kl write 1 atlev 0 adv)
!
mapx $WORK\f0 to $WORK\f1 using $cds\control.txt exp
lev 0 entry
!
u-sort $WORK\f1 to <to>
```

LBRIEF

To print out only the short caption (*u) and identity (*i) fields.

```
[from/a/r,to/k,gap]
print from [from] to [to$*] with

w 132
d d (a) k;

d *rec (a) g[gap$0] ;
d *i*is (a);
d *i1 (a) ':'; d *i2 (a) '-';
d *u (a) t20 1;
!
```

LIST

To list out (print) from the built records in a file.

```
[from/a/r,to/k,gap]
print from [from] to [to$*] with

w 58
e
d d (a) s x f s;
```

```
d *rec (a) g[gap$1] e (t) ' #';
d *i *store *c *num *t *k *m *z *n *ns *p *d *l *r *n
  (f) s x f s (il) ' *' f s;

d *i1 (a) ':'; d *i2 (a) z '-';
d *f1 *p1 *d1 *l1 *e1 *store1 *rp *is
  (il) s x f s;

d *fd *pd *dd *ld *ed *stored *rd
  (a) s '<';

d *p2
  (a) s '/';

d *g *prod *coll *acq *recr *ns *g0 *g1
  (a) g0 x f s;

d *f (f) e s (il) s x f s;
!
```

LISTE1

To create an appropriae bracketing system for records.

```
^from/a/r,to/k^
print from ^from^ to ^to$*^ with

w 58
c e
d d (a) s x f s;

d *rec (a)
           [ *i (f)
                    [ *is (a) k]
                    '*i R.' +
                    [ *is (a) + '.']
                    ' *store {|' + (il) s (t) '|}']
           [ *i1 (a)]
           [ *i2 (a) z '=']
           g1 e (t) ' #';
d *store *c *num *t *k *m *z *n *ns *p *d *l *r *n
  (f) s x f s (il) ' *' f s;

d *f1 *p1 *d1 *l1 *e1 *store1 *rp *is
  (il) s x f s;

d *fd *pd *dd *ld *ed *stored *rd   (a) s '<';

d *p2
```

```
  (a) s '/';

d *prod *coll *acq *recr *ns *g0 *g1
  (a) g0 x f s;

d *g (a)
           [ *i (f) '*store {\' (il) s (t) '\}']
           [ *is (a) + '.']
           [ *i1 (a)]
           [ *i2 (a) z '=']
           g0 x f s;
d *f (f) e s (il) s x f s;
!
```

LISTM

To list out files of marked records (suitable for editing).

```
<from/a/r,to/k/a/r>
print from <from> to <to> with
w 70
c
d L2 (a) g1 '*g1 ' ;
d L1 (a) g0 '{l=0 g0 \' + '\ l=10} ';
!
```

MAKEF

To set up a format for the data.

```
[ ]
makef $formats\cds from
\ cds data format - based on Museum object format
rec ( num i c store u m t k kp ke kl kd ka
      z prod coll acq g qv ns recr g0 g1 )
num = i    \ Record identity generated by discat
i ( is i1 i2 )  is = s  i1 = i i2 = i  \ identity number
c = s      \ general class of object
store ( store1 stored )  \ storage location
store1 = s  stored = s   \ main part and detail
u = s      \ simple name
m = s      \ medium (e.g. photograph)
t = s      \ full name or description
k = s      \ keywords
kp = s     \ person or people keywords
ke = s     \ ethnographic group keywords
```

```
kl = s    \ locality keywords
kd = s    \ date keywords
z = s     \ size (ignore if photo contains a ruler)
prod ( f p d e l r n )  \ production
coll ( f p d e l r n )  \ collection acq  ( f p d e l r n co )
\ acquisition
g ( i store u m t k kp ke kl z n )  \ info about parts
pt = s  \ title of part
qv = s    \ "see also"
ns = s    \ general notes
f ( f1 fd )  \ "function" word
f1 = s  fd = s  \ main part & detail
p ( p1 p2 pd )  \ person
p1 = s  p2 = s  pd = s  \ surname, forename & detail
d ( d1 dd )     \ date
d1 = s  dd = s  \ main part and detail
e ( e1 ed )     \ ethnographic group
e1 = s  ed = s  \ main part & detail
l ( l1 ld )     \ locality
l1 = s  ld = s  \ main part & detail
r = s         \ reference number
n = s         \ note
recr ( rp rd )  \ recorder
rp = s  rd = s  \ person & date

g0 = s g1 = s \ general text fields
ka = s        \ new field, put in 9.8.1989
co = i        \ new field, put in 9.8.1989
!
```

NUMBER

To automatically number a file of records.

```
<from/a/r,to/k/a/r,lettercode/k/a/r,startingat/k/a/r>
edx <from> <to> exp
openout 1
$ n setto *rec(*i/r(*is '<lettercode>' *i1 <startingat> *i2 0))
re ( read 0 to x
     $ x (mate n write 1)
     $ n (goto *i1 i add 1)
   )
!
```

PRINT

To print out from the built records in a file.

```
<from/a/r,to/k,width/k,opts/k>
copy $cds\dspec to $WORK\fd from2
w<width$70>
<opts>
!
print <from> to <to$*> with $WORK\fd
```

Q

```
To go into the 'q' or query system.
<on/k,opts>
q file <d->db pspec $cds\pspec with
t v '' <opts$>
p 'CDS>'
!
```

SETUPIR

```
To set up a Direct Access (DA) database from a built file.

<from/a>
c-index <from> to \muscat\work\fs
edx \muscat\work\fs \muscat\work\ft exp
openout 1
re(read 0 to x
   $x(lev 1 goto *i write 1 atlev 1) or (setf /0 write 1)
  )
!
u-sort \muscat\work\ft to \muscat\work\fr opt fo
DArec \muscat\work\fs file DArec with
i *is k1 n
!
DAterm \muscat\work\fr file DAterm with
k1 n
!
```

SORT

```
To sort a large batch of records.

<from/a/r,to/k/a/r,opts/k,size/k>
sort from <from> to <to> to2 &W0/<size$LA>
             work &W1/<size$LA> work2 &W2/<size$LA> with
<opts>
!
```

TERMSOF

To print out all of the terms of a single built record, to see what they would be in the database.

```
<from/a/r,to/k>
c-index from <from> to $WORK\f1
print $WORK\f1 to <to$*> with
d d (a) g0 t4 1 "'" + "'";
d *rec (a) g0 k;
!
```

UPDATE

To add records to a database when they have already been added in before and have only been slightly changed.

```
<from/a/r,style/k>
c-index<style> <from> to $WORK\f1
DBadd from $WORK\f1 file <d->db with
n p i *is
!
```

VOC0

To produce a diagnostic output of everything except the long text (*t) and notes (*ns) fields.

```
{from/a/r,to/k/a/r,cutoff/k}
keyx from {from} to $WORK\f3 exp
goto *i write 1 atlev 1
re ( (code *p or string()
      not(code *z or code *t or code *n or code *ns)
      write 1 atlev 0
      adv)
     or next )
!
u-sort $WORK\f3 to $WORK\f4 opt fo
edx $WORK\f4 $WORK\f3 exp
openout 1
repeat ( read 0 to x
         $x (lev 1 $c setto *i2 0) or ($c i add 1)
         $c i try ( (ls {cutoff$11} $x write 1) or
                    (eq {cutoff$11} $*stored '...' write 1) )
```

```
       )
!
print $WORK\f3 to {to} with
w40 c
d L1 (a) g0 x s;
d L0 (f) l=8 g0 (il) s (t) l=0;
d *p1 (a); d *p2 (a) '/'; d *pd (a) '<';
d *is (t) ':';
d *i2 (a) z '-';
!
```

APPENDIX S

THE DISCATEL MACROS

This contains 45 macros. A number of these are identical to macros in the cds and el directories. The following are macros which are not identical.

ADD

To add a file to the database.

```
<from/a/r>
c-index <from> to $WORK\f1
keyx $WORK\f1 to $WORK\f2 exp
(lev 1 goto *id write 1 atlev 1) or (setf /0 write 1) !
u-sort $WORK\f2 to $WORK\f3 opt fo
DBadd from $WORK\f1 from2 $WORK\f3 file <d->DB with
p i *ids
!
```

BATCHDEL

To delete a batch of files from the database.

```
<from/a/r,startingat/k/a/r>
c-build <from> to !a
c-delnumber !a to !b startingat <startingat>
DBadd from !b file <d->db with
dp i *ids
!
```

BUILD

To build a muscat record.

```
[from/a/r/h,to/k/a/r]
build from [from] to [to] with
i *id *ids . *id1 : *id1 - *id1 = *id1 #
!
```

CREATE

To create a database.

```
<kilobytes/a/r>
DBcreate <d->DB bytes <kilobytes>000 blocksize 6144 with
k1
!
c-build <d->intro to !R
c-add !R
```

DADISC.TXT

To enter a Direct Access (DA) database.

```
<with/k,opts/k>
copy to !discat-spec0 from
<opts>
!
copy to !discat-spec from !discat-spec0 from2 <d->dspec from3
<with>
-
!
<with/k>
disc terms <d->daterm recs <d->darec pspec pspec with !
discat-spec
delsf !discat-spec
delsf !discat-spec0
```

DEL

To delete a single record from the database.

```
<recnum/a/r>
c-build to $WORK\f1 from
#H *id *id <recnum> #
!
dbadd from $WORK\f1 file <d->DB with
d
i *ids
!
```

DELNUMBER

To delete a set of numbered records from the database.

```
<from/a/r,to/k/a/r,startingat/k/a/r>
edx <from> <to> exp
openout 1
$ n setto *rec(*id/r(*ids 'R'
               *id1 <startingat> *id1 0))
re ( read 0 to x
     $ x (mate n write 1)
     $ n (goto *id1 i add 1)
   )
!
```

DISC

To enter a database (DB).

```
<with/k,opts/k>
copy to !discat-spec0 from
<opts>
!
copy to !discat-spec from !discat-spec0 from2 <d->dspec from3
<with>
-
!
disc file <d->DB pspec <d->pspec with !discat-spec
delsf !discat-spec
delsf !discat-spec0
```

EXTEND

To extend a database.

```
<kilobytes/a/r>
DBextend file <d->DB to <d->DB2 bytes <kilobytes>000
msdos del <d->DB.da
msdos ren <d->DB2.da DB.da
```

INIT

To initialize the system and set aliases.

```
||
alias c $macros\discatel\
alias f $macros\discatel\f
alias a $macros\cds\
format \muscat\formats\discatel
```

```
alias d ''
```

INDEX

To index a set of records.

```
<from/a/r,to/k/a/r>
indexx from <from> to <to> with <d->ispec exp <d->iexp
```

LIST

To print out a set of records, with codes.

```
[from/a/r,to/k]
print from [from] to [to$*] with
w 58
c e
d d (a) g0 x f s;
d *rec (a) g1 e (t) ' #';
d *ids (a);
d *id1 (f) '.' (il) z '-';
\ put any other d-directives here
!
```

LOADR

To load records from a DB to a DA database

```
<from/a/r,to/k/a/r>
DArec from <from> file <to> with
i *ids k1 n
!
```

LOADT

To load terms from a DB to a DA database

```
<from/a/r,to/k/a/r>
DAterm from <from> file <to> with
k1 n
!
```

MAKEF

To make a format.

```
[] makef !F from
rec ( \ put here the list of fields which are declared
      \ below. But not ids and id1, which are subfields
      \ of id. The order in this bracketed list is
      \ unimportant.

      id
      head text
      a b c d e f g h i j k l m n o p q r s t u v w x y z
    )
id ( ids id1 )  ids = s  id1 = i \ identity number
head=s   \ general heading for the record
text=s   \ general field for text

\ data fields: alter or extend as necessary

a=s b=s c=s d=s e=s f=s g=s h=s i=s j=s k=s l=s m=s
n=s o=s p=s q=s r=s s=s t=s u=s v=s w=s x=s y=s z=s

\ remember that the order of declaration a=s to z=s
\ determines the order in which DISCAT will display
\ the records on the screen. Also remember that you
\ can only add new fields at the end of this list
\ unless you are prepared to throw the DB file away
\ and recreate it.
!
```

PRINT

To print records.

```
<from/a/r,to/k,width/k,opts/k>
copy <d->pspec to !pspec from2
w<width$70>
<opts>
j \
!
print <from> to <to$*> with !pspec
```

RELOAD

To make a DA database out of a DB database.

```
<>
c-unloadt  db.da       to  $work\f1
c-loadt    $work\f1   to  daterm
c-unloadr  db.da       to  $work\f2
```

```
c-loadr    $work\f2   to  darec
```

SORT

To sort a set of records.

```
<from/a,to/k/a,fields/k/a,kill/k,opt>
skey from <from> to $work\f1 with f <fields>
k <kill>
!
sort  from  $work\f1  to  <to>  to2  $work\f2  work  $work\f3  work2
$work\f4 with
<opt>
!
```

SYSTEM

To use the macros and files in another (specified) directory.

```
<directory/a/r>
alias d <directory>\
```

APPENDIX T

A QUICK START TO THE CDS INTERACTIVE SYSTEM

(Whenever you see ENTER, please press the 'Enter' (return) key. When you see ESC, please press the ESCAPE key.)

Set up a sub-directory called 'database' (or some other name) off your 'root' or top level of the hard disc

go into the newly created directory

type the letters: cdsi

select the "new" option by pressing the ENTER key

type in a name (for instance your own name) ENTER

press ENTER (which will give you 'yes' in the Select field)

move to right with right-pointing arrow ENTER

type in a name for the field, for instance 'title' (you can use the backspace key to correct mistakes) ENTER

move with right-pointing arrow to 'Indexing' and ENTER

move down to 'Free Text Mode' and ENTER

move to the 'Name' field of *b and ENTER

type in 'author' and ENTER

press ESC   (you will see the relevant 'macros' being written)

select 'Change size' by pressing ENTER

type the number 50 and ENTER

select 'Create Database' and ENTER
 select 'Edit Record Source Files' and ENTER

type a filename, for example 'test' and ENTER

type in five or six words (the title of a book)

press F5

type an author's name

press F3

(you can repeat the steps above if you want one or two more records in your trial)

press ESC

select 'Add Records to Database'

press ENTER on the name of your test file

press any key when asked to do so (do not worry about the message 'No Data Files Found')

move to 'Use Database' ENTER

press the letter 'f' (for free text search)

press the letter 'f' again

type in one of the words you used in the 'title' field (you can use the backspace key if you make a mistake)

press the ENTER key

press the letter 'g' (for 'go to first record') (you should see one of the records you have created)

press the letter r twice

press the letter x

select 'Exit' with the arrow key and press Enter

press ESC twice

(To clean out what you have done, delete all files in the directory and sub-directory which you created; the directory will be apparent from the name you gave it, and it has a sub-directory called 'Data' which will need to be emptied and removed.)

APPENDIX U

TWO WAYS OF SEARCHING FOR A SET OF RECORDS AND PRINTING OUT

You can search through for a whole set of records (this only works on R records), outside the database system. For instance, if you wanted to print out all the records containing a certain date or word or combination of these, without having to look at each and mark it, you could do this.

In order to do this, use your word-processor to create a small file with your query in. This query must conform to the various terms used in the database system. For instance, you might put in a query:

'P=Mills' and 'M=photographs' and 'D=1922'

which would find all the photographs taken by Mills in 1922.

Or you could put in:

'M=photographs' and 'shield'

When you would have all records of photographs in which the word 'shield' was used.

Or you could put in:

'shield' or 'spear'

When you would get all the records with either the words 'shield' or 'spear'.

The words used here must be the exact, 'suffix-stripped' terms in the index. If you are not sure about whether a word is suffix-stripped or appears in the index, it is easy to find out. When in muscat, go 'c-q'. This takes you into the 'q' or query system. Now type the letter i followed by the word you are interested in. You will be taken to the nearest equivalent. For instance, type

i boy

and you will receive the reply 'boi'. You can look through the list by pressing the carriage return. When you want to leave the 'q' system, type 'stop' and you will return to muscat.

Once you have created a little file with your query in, which must have the extension .txt, you can use it. Supposing you wanted to search for all records of Mills photographs in 1922, as above, and you called the file 'Mills.txt', you would then go, in Muscat:

c-getrecs

You will then be prompted for the query, to which you reply 'Mills', as follows:

query=Mills

You will then be asked for a file to which the answer can be put, to which you could reply with a file name, for instance Mills2.

The search will then be done and any records found will be put in Mills2.mus. In order to see the results, this needs to be turned into a text file, by going:

c-print mills2 to mills3

There will now be a text file, mills3.txt, which you can print out above, either with a word-processor, or by using the MSDOS 'print' command.

A variant of that above, is to follow the method for finding and printing records using 'q', which is explained in the Muscat Manual, 3rd edn, p.156. Specifically, you would input a Boolean or free text query, then use 'mto' (output the matching documents to a file), and print out the results.

APPENDIX V

THE CAMBRIDGE DATABASE SYSTEM INTERACTIVE (CDSi).

This appendix describes a stand-alone, screen-driven version of the system which will allow you to set up your own database specifications and enter data. If you would like to see quickly how this works, follow the instructions in Appendix T.

In carrying out the suggestions below, please use the arrow keys to go up and down lists of choices (menus), and the 'Enter' and 'Escape' keys, as instructed, to make choices. Whenever you need relevant help, press the F1 function key.

GOING INTO THE DATABASE

Set up a directory off the root of your computer, called 'Data', or some other name. (You may find that one has been set up for you already.) Go into that new directory and type the letters cdsi. (You may have a batch file already to do this, so try typing the two leters db in case).

You will find a screen with 'Choose a Database or Esc. to Quit', and 'F10 to change defaults' at the bottom. Press the F10 function key and you will be able to change the default settings for all your databases. Press the F1 key (Help) for an explanation of what each setting means. You can leave these settings for the present, altering them later if necessary.

Return to the previous menu by pressing 'Esc'.

CREATING A STRUCTURE

If there are no current databases already set up, or you want to set up a new one, select "New" by pressing the Enter key when "New" is highlighted. (Throughout, you can move around the screen using the arrow keys on your keyboard). If you have already created other specifications for other databases, one of these, possibly with modifications, may be suitable for your new database. In that case, select the name of that specification.

It is possible at any time to revise a specification if it is found to be inadequate.

You should then give your new database a name. We shall be using the example of a database of video cassettes, so you could use the name 'video'.

You will then be presented with a screen allowing you to define the indexing and screen printing structure for your database. These are known as 'specifications' and set up the ways in which records are divided into fields, how each field is to be indexed, and how each field is to be printed out on the

screen.

An example You may like to try out setting up options using the following example, which would be the sort of scheme one might have for a database containing video-tapes.

Code	Select	Name	Indexing	Display	Cap	Field
*c	yes	title	B	0,P,L,N	Cap	
*d	yes	date	D	2,T,C,I		
*e	yes	area	S	2,T,C,I		
*k	yes	keywords	F	2,T,C,I		
*l	yes	country	S	2,T,C,I		
*q	yes	length	-	2,T,C,I		
*t	yes	series	S	2,T,C,I		
*u	yes	content	F	2,T,C,I		
*w	yes	reference	S	2,T,C,I		

Code The code (for example *c) is automatically set; you do not need to alter this. It indicates to the computer that a new field is starting. There are 26 codes or fields available. Since the record is printed on the screen in code order, it is sensible to start with material which you will want to have at the top of the screen page.

Select You can explicitly select a field by pressing the 'Enter' key when the cursor is in this column. To de-select it, press 'Enter' again. If you give a field a name, it will automatically be selected.

Name Give each field that you want to use a name, for example you might use words such as author, title, date. To type in the name, select the field by pressing Enter and then type in the name. You can use the backspace, arrow and delete keys to alter this if you want. You can come back and change this at any time.

Once you are satisfied with the name, press Enter or Escape and you will see the name on the screen. Move on to the next column with the arrow keys.

Indexing Select the options by typing Enter and you will be given the following options, each of which can be altered by pressing Enter on it (a toggle switch between yes and no).

Structured Mode: for 'and/or' queries. This is useful if the field is likely to hold a single word or a few words at the most (for example person, place, date, short title). It can be used

on its own or combined with the 'free text' form of indexing.

This field must not contain more than a maximum of about twenty words. If you are putting a record in through the on-screen editor, then you will be prohibited from putting in records of over that length.

Free Text Mode: useful for the above, but also for a string of words, for instance a short description consisting of one or a few sentences, every word in which will be indexed (except for words such as 'and, the, of').

It is not advisable to index fields which contain more than a paragraph of text. The upper limit is set by the fact that a whole record cannot contain more than about 500 indexing terms. If you try to index too many words, an error will be reported and you will need to shorten the record. (Further advice on this is contained in the Manual, Appendix D, where it is shown that the number of words you can index is variable, depending on the 'blocksize').

You will learn by experience when to use the two modes of indexing and when to combine them. As a preliminary rule, it is worth using 'structured' alone for dates, names, places and cross-references to numbers (for instance archival storage). If you are describing something in a line or few lines of text, then use the free text mode on its own. It is worth indexing short titles (for instance a one-line title of a book), in both ways.

Date mode: for dealing with dates, days, months and years and spans of dates.

Making your choice.

You can go down this list selecting your choice for each field. When you have made each selection, type Esc and you will see what you have chosen (S=Structured, F=Free Text, B=Both D=Date). You can return and alter these while this screen is still present by moving up and down with the arrows. You can also revise the specification later if it is not satisfactory.

Display options

Again use the arrows to reach a choice, and Enter to make a selection.

Separation: in other words, line separation between fields. You can select 0,1 or 2 with the Enter key. Normally, 2 looks best, except with the 'Caption' field, where 0 looks best.

Plain/Title mode (a toggle). The name or title of the field is printed first if Title is selected. It is normally best to use the 'Title' mode for short fields, for instance names, places etc., which are also the ones you will search by structured queries. The 'caption' field, and the longer text fields, indexed by 'free text' mode alone, or not indexed, look better in 'Plain' mode.

Line/Comma separation of repeated fields; each repeated field either being put on a new line, or separated by a comma. In general, short fields (for instance names or places), look best separated by a comma. Longer text fields look better separated by a line.

Indent/No Indent - indent or do not indent the text from the left. Short fields look best indented; longer text fields are better not indented.

The defaults, as set on the screen, are usually the best choice. If you try to change the caption field values the program complains, but will let you do so on confirmation.

Cap Field allows you to decide which the caption field will be. Only one field can be set in this way. One of the ways of retrieving your data is by 'caption mode', which lists the records by their caption fields (one of which can be chosen to take you to your record). It is a quick way of looking through a set of answers. Choose a field in which you have a line or so (for example, the title of a book or short description, which will usefully appear in a list). It is wise to leave the default values set for this field, but you can change them if necessary.

You should set up the fields as you wish. If you want a field to be used, but do not want the words in it to be indexed, give it a name but set all the indexing options to 'No'. One or two fields of this type for your notes, or for long texts which you do not want indexed, may be useful.

A record corresponding to the indexing format which you have now set up would look as follows:

```
*c Peoples of the Desert
*d 1961
*e Africa
*k war
*l Kenya
*q 50 mins
*t Tribal Worlds
```

*u A film about the nomadic tribesmen on the border of Kenya and Uganda
*z 204.6
#

Once you have set up your desired format for the particular database and checked it is as you want it, type 'Esc'. The various indexing and printing programs will then be automatically set up for you in your database directory.

If you want to change a specification after you have made it, select the 'Change specification' option on the menu. Select the database which you want to change and edit it. (When you have already set up other databases with their specifications, you can copy the specification of one of them as a model and modify it for your present use.)

In order to make the best use of the database system, it is important to have thought out how you would like the indexing and field structure to be. Try it out on one or two records, and revise it, before entering too much data. Otherwise you may have to re-enter or modify your data, which is always time-consuming work.

CREATING A DATABASE

After the indexing format has been created, you can create a database. When you enter the selection and choose your new database name you will see various options.

If you select 'create database', an empty database of 100k will be created. If you want to extend this later, you can do so; or the system will automatically increase the size if you run out of space while putting in new records.

If you want to start with an empty database of another size, larger or smaller, select 'Change Size' and specify the number of kilobytes (k) that you would like. Then select 'create database'. It is probably best to start with the default database size (100k), though later you may want to use slightly smaller or larger databases.

Once created you can select the new database you have just set up and you will be offered various options.

ENTERING DATA

There are three ways of entering data. The simplest is to select the 'Edit record source files' option system and follow the instructions. This will also prevent errors in coding.

When you select 'Edit record source files' you will find that the screen is split horizontally with a "window" above showing the contents of the current record. The "window" below is where the field editing takes place.

The top 'header' gives the database name and the current record number (i.e.. sequential number for the edit, not the database record number which is added automatically when the record is added to the database); the mid-line 'header' shows the characteristics of the current field. At the bottom of the screen are a list of the function keys and what they can do.

Using the appropriate function keys you can enter data into a field. You can move to the next field with F5, and to the next record with F3. You can move back to the previous field with shift/F5 and to the previous record with shift/F3. You can repeat a field with F6, and delete a field with F7. (You will be able to delete a record with F , later; and go to the end/start of the file with F . You will also be able to set 'constants', that is fields which are automatically added to each subsequent record, until they are cancelled.) When you have finished editing a set of records, type 'Esc' and you will be asked if you have finished. Type the letter 'Y' for yes, and the file will be stored away with the name you have given it. You can make alterations, or add further records later by selecting the file name again.

Remember never to use a backslash (\ ) in your text within the editor, this is a reserved character and will corrupt the record in various ways.

ADDING RECORDS TO THE DATABASE

Select 'Add Records to Database'. You will see the file name or names of files ready to be added. Select the one you want added with the Enter key and it will be added. If the file is fairly long and a lot of indexing is being done, this may take some time. A complex indexed file is set up which will make retrieval very fast and powerful.

Once the file has been added to the database, the name will disappear from the screen (though the file itself is not destroyed and can be recovered, as explained later).

ENTERING AND USING THE DATABASE

Select 'Use Database' and wait while the system is loaded. You will then be presented with options. These are explained in the various 'help' options on the screen, and in chapter four of

the Manual.

PART C: TUTORIALS

TUTORIAL ONE. PREPARING RECORDS AND LOADING THEM INTO A DATABASE

A way of simply adding records to a database is given in the CDS Interactive Manual. The following account is for those who want to progress beyond that to deal with more complex data.

EXERCISE 1

Preparing, checking and numbering a test record.

In order to see how the system works, let us go through all the stages with a simple record. Since you will begin to create a good many files during the processes explained in this Tutorial, it is worth remembering for later that you can create 'temporary' files in cds 2000. These are automatically deleted when you leave the 'Muscat' program, hence saving the need to clean up after you. These temporary files start with an ! mark. For instance, you could use !a !b and so on. In the following exercises, however, permanent file names have been suggested. This is because the temporary files are kept in 'core', that is in RAM. Unless your files are fairly small, or you have plenty of free RAM, you will run out of space and get a message saying 'space exhausted....'. You should use the permanent names in the exercises and then clean up after you by deleting the unwanted files you have created.

Typing in the record.

Using your usual line editor or word-processing package, prepare to type in a few lines. Note that these must not include the 'control' characters (automatic line endings etc.) which are produced by a number of word processing packages. To avoid including these illegal characters, if you are using a word-processing package and there is a choice in the package at the start, use the 'non-document' mode. If the choice comes at the end, then choose the 'non-formatted' option when you file or archive the document.

In order to go into the database, the file needs to be stored in the 'cds' sub-directory of the Muscat directory. Go into that sub-directory. It also needs to have the extension '.txt'. Think of a name for your small test file and start to create a file called 'test.txt'.

Now type in a short record consisting of five fields as follows:
*kd 19.10.1987

*kp Marx/ Karl
*kl Cambridge
*u a visit to Cambridge by a north London witchcraft coven in October 1987
*t This is a detailed account in the local paper of the visit of
the famous north London witchcraft coven (this could go on for up to fifteen pages) #

This is enough for a test. It contains a date, name, and place keyword fields, and a shorter and longer text. The record ends with a hash sign. The fact that a field is being declared is indicated by a star; the type of field is indicated by the letters after the star. Now save this file (in a non-formatted form, as stated above) and return to the 'cds' directory, where you should find this small file. Check that it is there and in the form you typed it in, without any control characters.

Checking the record is suitable for input.

In order for the record to go into the database, it has not only to be 'clean', but also 'built' into a form that the database will accept. In order to check its accuracy and prepare it for inclusion, type the following (make sure you are in the \muscat\cds directory before doing this):

muscat cds

This will take you into the muscat system, giving you a prompt with the word muscat. (Whenever you want to leave the muscat system, type the word 'stop'.)

Now build and check the record by typing:

c-build test to test

(or whatever word you have chosen to call your file). The full form of this would be:

c-build test.txt to test.mus

The computer assumes the extensions, so it is not necessary to type them in. If the program cannot find the named file, it will complain. Otherwise you should get a report telling you that a record has been built.

If there are any errors in the record, you will be told, with some guidance as to which line the error was in and what kind of error it is. (Some help with error checking and

line-finding is given in appendix B.)

When you have done this exercise, it would be worth using the same record but modifying it slightly (for instance by typing *kz instead of *kp, which the program will not recognise), in order to see the kind of error message you will get. These errors have to be corrected before the record will build and can be entered into the database.

Numbering the record.

Once a record is clean and in a built form, it is ready for numbering. All records have to be numbered before inclusion in the database. You do this by typing:

c-number test to test1

You will then be asked for a letter code. Type a capital R in reply (standing for Record). You will then be asked for a number to start at, which would mean that a whole file could be automatically numbered. Type 1000 in reply to this request. Your record will then be given the number R.1000.

EXERCISE 2

Setting up a database and adding the record.

Let us assume that you do not have a database as yet. (If you do have a database, or someone else has created one, be careful as you could over-write it. Make sure by looking to see if there is a file called db.da in the \muscat\cds directory. If there is, it is a database file and should be renamed temporarily so that it is not over-written or destroyed.)

To create an empty database go into the '\muscat\cds' directory and into muscat (as explained in exercise 1) and type:

c-create

You will be asked for the size of the database you want to create, in kilobytes (1000 bytes), so respond by typing 200. You will be told when the database is ready. You will see that it already contains some records and is partly full (a percentage is given); these records are the introductory menu and help pages which you will need and which are automatically put in. You can alter them if they are inappropriate, as explained in a later exercise.

Now add the test record you have created in exercise 1

by
typing:

c-add test1 (or whatever name you chose)

This record will be added to the database and you will be given a report on how many index terms have been extracted, how full the database is, what version of the database is now active etc. The database is now ready for searching.

EXERCISE 3

Searching the database to see your sample record.

The ways to search the database are described in a separate part of the manual, which you may want to read now. But if you would like to read the full manual later, and just want to see the result of putting in your sample record, type the following:

muscat cds

At which you should get a 'Muscat' prompt. Then type:

c-dbsys
(If the machine complains that there is no videodisc, then try typing c-dbsys opts 0).

Each of these routes should take you into a screen which welcomes you to the system and gives you a menu of choices.

On the keyboard, type the letter 'f' (thus selecting 'free text query'). You will then be asked to put in a word or words.
Type the word 'ghost' (if you make a mistake, just use the backwards pointing arrow at the top of the keyboard, which will delete the letters you have typed, or try again after a carriage return). Then type a carriage return. Then type the letter 'g'(for 'Go to first item') and your sample record should come up on the screen.

Alternatively, type 'R', which will take you back to the initial menu and type the letter 's' (for structured query). Then type 'l' (for locality), and a carriage return. A list, simply containing the word 'Cambridge' should appear. Type a carriage return to select this and Cambridge will then appear on the right-hand side of the screen. The query has been set up. Type the letter 'r' twice to return to the

screen which has 'Go to first item' on it and type 'g'. You will then be taken to your sample record again.

To leave the database, type the letter 'r' until you come to a screen which says EXit at the bottom, then type the letter x.
Then type 'stop' to leave muscat.

Now try creating and numbering two or three more records, using the same fields, but putting in some data of your own choice. You will then see how the list of possible places, persons etc. increase in size. You will also notice that the *t field is not indexed. Remember to end each record with a hash sign, otherwise the records will be treated as one record.

If at any time you want to get rid of your trial database, you can delete it like a normal file by typing:

```
del db.da
```

(specifying the cds directory of course, if you are not in that directory).

EXERCISE 4

Preparing a more complex videodisc record.

The simple example above, with some additional fields, will serve for many text database purposes. But if the system is being used to link with optical media (videodisc etc.), it has some extra features, which can be illustrated through an exercise.

Using your word-processor or line editor as before, create a file (or edit your \cds\test.txt file); add a new record as follows:

```
*i B.47000 *kd 14.12.1983
*kl Oxford
*k sport
*u the Oxford rugby football team
#
```

This record has a *i field, that is an index field, which cross-refers to a still image (denoted by B.) at frame 47000 of the videodisc. Save this file in the \muscat\cds directory, as before, then return to that directory and go into muscat by typing:

muscat cds

Then check the record, as before, with c-build. When it is clean, you are ready to proceed.

The *i number needs to be converted by the computer so that when it is retrieved it will be displayed with a 'Show' box at the bottom of the screen which will take you to the videodisc frame. This is done in one of the programs within a command called 'batchadd'. So type:

c-batchadd

You will then receive prompts for filename (give the test file name), and, as before, for record type (type 'R') and the number to start at (type 1100). The rest is done automatically, and you are told of the progress of the program.

The new record will now be in the database. Now type:

c-dbsys

which will give you a simulation of what you would see.

Once you are at the introductory screen, select 'f', then type 'football' and carriage return. Then type 'g', for 'Go to first item'. The record should now appear with 'Show' at the bottom of the screen.

Type 's' and frame 47000 will be displayed (which is unlikely, of course, to be an Oxford football match), or a message saying frame 47000 is on show will appear, if there is no videodisc player attached.

To leave the database do as before, i.e. keep typing 'r' until you are at the introductory page and then type 'x'. And type 'stop' to leave Muscat.

EXERCISE 5

Preparing a text record.

For many applications, users will be content with texts appearing in the *t field of a record. This is perfectly adequate where the texts are short and there is no desire to read through the set of texts sequentially, as if they were a book.

But if the text is the most important part of the record, for instance if you are using the database as a way of holding and retrieving from a number of book-length texts which you want to read through sequentially, as well as finding as a specific record, some extra processing needs to be done. Let us try a small dummy sample.

As before, use an editor to create a file called something like book.txt. Then type the following (for now the fields should be typed in twice; there are automatic ways to do repeated fields, by setting constants, as explained in the manual, if you need to do longer texts);

```
*c Pride and Prejudice
*m book
*prod *d 1798
*kp Bennet/ Mrs
*prod *p Austen/ Jane
*u marriage and fortune
*t  It is a truth universally acknowledged,  that a single man in possession of a good fortune must be in want of a wife. #

*c Pride and Prejudice
*m book
*prod *d 1798
*kp Bennet/ Mrs
*prod *p Austen/ Jane
*u marriage strategies of neighbours
*t  However little known the feelings or views of such a man may be on his first entering a neighbourhood,  this truth is so well fixed in the minds of the surrounding families, that he is considered as the rightful property of some one or other of their daughters."
#
```

You now need to split this into the two R-record and T-record files, the one providing the index to the other. You do this as follows. Go into the \muscat\cds directory and then into muscat.

```
c-build book to book
c-number book to book1     (this numbers the file)
```

At this point, you are given a request:

```
lettercode =   to which you reply T ('T' in upper case).
startingat =   to which you put a number at which the
               numbering of the records must start; this
               must not overlap with any other R or T
               numbers already used. Since the introductory
               have T records, you should start at 500.
```

When this is completed, you are in a position to split the file by going:

c-bksplit book1
You will then be asked for names for the two files which are created, to which you can answer as below:

tor = book1r (tor, i.e. to record file)
tot = book1t (tot, i.e. to text file)

You now have the two separated and numbered files. These still need some global editing before they are ready to go into the database.

Editing the new record file (book1r)

You first turn this back from a built file into a text file by going:

c-list book1r to book1.tor

Then you use whatever editor you have and do a global edit of book1.tor, as follows:
globally exchange *i R to *store {|T.!|} *i R

The effect of this is to automatically put in the cross-references to the related Text records.

You should also globally exchange *t to *g1. This will put in a print command to create a space between each paragraph of text.

Editing the text record (book1t).

Again you need first to convert this to a text file:

c-list book1t to book1.tot

This needs then to have the cross-references to previous and next records put in. Book1.tot is globally edited as follows:

globally exchange *i T to *store {|0 T.!-1 T.!+1|} *i T

This will automatically set the links between all the paragraphs of a book or diary, however many there are.

It is necessary to slightly modify this for the first and last records, which will apply to the example you have typed in

above. The first paragraph will not have a previous paragraph and hence should be edited to:

*store {|0 0 T.!+1|}

The last paragraph should be edited to:

*store {|0 T.!-1 0|}

The file is then ready for entering into the database.

Entering the files into the database.

The procedure for entering files into the database is explained in detail elsewhere. Here we may note two things. Firstly, that in order to enter these into the database,they need to have the extension .txt. So the two files might be named or renamed, for instance,

bookr.txt (the record file)
bookt.txt (the text file)

Secondly, these have now all been numbered and cleaned up in various ways. They do not, therefore, go in using the procedures in the normal BATCHADD command. Rather they are added with a simpler command.

They need first to be built from the text files, thus for instance, having entered muscat cds, type:

c-build bookr to bookr

Then this file is added thus:

c-add bookr

Likewise with the bookt.txt file.

When you have a clean file, to see the effect in the database do the following.

Go into the database in one of the ways which you have already used in previous exercises.

Type 's' for structured and 's' again; then 'p' for person and 'Enter' and select 'Austen' in the list of persons by pressing the carriage return on the box against that name. Type 'r' until you get back to the selection menu with 'Go to first item'. and then type the letter 'g'. You will then have the record associated with these texts. Type 's' for show and

you will see the text with the caption you have put in.

If you want to see the next paragraph, type 'n' for next ('p' for previous will then appear to take you back). As you can imagine, with a longer text, you would be able to go to any paragraph, if it contains appropriate index terms, and then read forwards or backwards. When you leave the text in this mode, you will always return to the record by which you entered.

EXERCISE 6

Altering the introductory screens.

The database contains an unlimited number of data records, but also a user-interface or front end which you can modify to your own use. The material in the front end is contained in a set of records which are in a text file in the \muscat\macros\cds directory called 'intro.txt'.

This file is automatically added when you set up a new database using c-create, as we have seen. It can be altered and then re-added at any time, the new version replacing the old one without disturbing the database. Thus if you want to add new choices, new help pages, new tutorials or whatever, this can be done at any time.

At present, intro.txt contains a number of A and T records which give you choices. You will find the text of intro.txt is given in Appendix K above, which is worth looking at in conjunction with this exercise.

Let us make a small modification to show how this can be done. Using your editor, get the file \muscat\macros\cds\intro.txt. Go to the first record, which starts *i A.1 and after the first line:

*g1 Welcome to the Cambridge Database System

add a line, for instance as follows...

*g1 Adapted for use by (add your name here).

Now save the file, go into muscat, build the file and add the file (as explained in previous exercises) and go into the database. You should now see your name on the introductory page. Obviously, if you want to change it back, you do the same thing in reverse, deleting that line and re-building and adding the file intro.txt. (An alternative way would be just to copy or set up a file with the one record *i A.1 in it and

add this.)

TUTORIAL TWO

HOW TO FIND DATA IN A DATABASE

INTRODUCTION

The following exercises will take you through all the types of retrieval that are possible with the retrieval system. If you do not have a videodisc attached, you will only get a simulation of the pictures or sound.

The examples are taken from a very small trial database which has automatically been loaded into your computer with the software system. This is held in two files called DAREC.DA and DATERM.DA in your \muscat\cds directory, comprising about 60k in all. There are the introductory 'menu' and help pages and 19 sample records of various kinds in this database. These records are printed out in full in Appendix J. You may like to use that Appendix later to work out some further queries based on the records. Clearly this is a very small set of records. You will have to imagine, before you have built your own database, how the system would work with far larger sets of records.

EXERCISE 1

Making choices.

This will take you through the various ways of making queries with the computer, showing each page as it appears on the screen, and then explaining what happens when you select a choice.

Go into the \muscat\cds directory and type:

c-disc

Look first at the screen, and note the 'choice' boxes. These choice boxes will be shown in this written description by using a * symbol. Whenever you see a *, imagine that it is a box on the screen.

Now look at the screen again, and you will see the following page appear:

(computer screen)

Welcome to the Cambridge Database System

Press '**H**' for help: *
Introductory text: *
Tutorials: *
Contents: *
Free text query: *
Structured query: *
Both free text with structured: *

Now press the 'H' (or Help) key on the computer and you will see a detailed description of how to make selections. Read that description carefully. In summary, the description explains that:

Each screen you see contains several little boxes like this: * * one of which will be flashing, or highlighted in some other way. You can change which box is flashing by pressing the four 'arrow' keys. If you press the 'Enter' key (the large key which is located where a carriage return key on a typewriter would be) you will select the flashing box.

A box is often accompanied with a word beginning with a letter in a special colour or shade, **L**ike **T**his. Pressing the letter key selects the
box, for instance '**F**' will select '**F**ree text query'.

As suggested on the page you see on the screen, if you press 'R' for Return, you will be taken back to the initial page. This is important to remember. Whenever you want to 'return' to a previous higher level, press 'R'. If you continue pressing 'R' you will always return to this initial page. In other words, choosing boxes, or highlighted letters will take you downwards and sideways through the system, choosing 'R' will always take you back up.

Be careful, however, not to press 'R' or any other letter for too long or too heavily, as the key is likely to 'repeat' and take you further than you intended to go.

Now try out the choice boxes and selecting by a single letter (which can be a capital or non-capital, in other words 'T' or 't'). For instance, have a look at the 'Contents' option. Select this by pressing 'c' on the keyboard, and you will then have a

list of choices with boxes. Choose one of these by moving to it with the keyboard arrows and pressing the 'Enter' (carriage return key). Then return back to the original page by typing 'R', which will take you up one level, and then typing 'R' again, which will take you to the introductory page.

You are now in a position to make choices, going up and down the levels of the system.

EXERCISE 2

Free text queries.

Let us now look at an example of what we call the 'base' page. Try pressing 'F' for free text query, and the following will appear on the screen:

```
_____________________________________________________________________
_
(computer screen)

* Free text query
* Input Marked item
* Alter the retrieval style
* Help
* Return
_____________________________________________________________________
_
```

By selecting 'Free text query', either by pressing 'f' or selecting the box, you will be able to enter a free text query. You may put in up to a line or more of words, but it is best not to put in too many; three or four words is often ideal. They can be in any tense, plural or singular, capital letters or ordinary letters etc. You can put them in any order, with complete freedom.

For instance, you could type 'Show me all the hats made with yellow orchid stems'. But since the really important words are 'yellow' 'hats' 'orchid' 'stems', it would be sensible just to put in these four words, thus saving typing and saving the computer from looking for 'show' 'all' 'made' (unless these are indeed important). When you have typed in a query, it is fed in by pressing the 'Enter' (carriage return) key.

Now press 'f' for free text query and you will see the following prompt:

```
 Free text query>
```

Try typing in a short free text query, for instance the following : hats yellow orchid stems.

This will appear on the top of the screen, after 'Free text query> '. If you make a mistake in the typing, it can be corrected as follows. Using the 'backspace' key on the keyboard (which often has an arrow pointing to the left, and is always situated just above the 'Enter' (carriage return key) you can delete letters back to the point where you want to re-type them.

You will now have a screen as follows:

```
_____________________________________________________________________
_
(computer screen)

Free text query > hats yellow orchid stems
* new Free text query
* Inspect the free text query
* Go to first item     * New item
* Input Marked items
* Alter the retrieval style
* Help
* Return
_____________________________________________________________________
_
```

Type an 'Enter' and now try typing the letter 'i' for 'Inspect the free text query' and the following screen will appear:

```
_____________________________________________________________________
_
(computer screen)

Del: *   freq   1 hat
Del: *   freq   1 yellow
Del: *   freq   1 orchid
Del: *   freq   1 stem

-------------------------------------------------------------------
-
```

This shows the words you are looking for, and shows how often (freq stands for frequency) they appear in all of the indexed material in the database. In other words, 1 record is indexed by the word yellow etc.

You may want to delete a record from a particular query, for instance if it appears so many thousands of times that it is unlikely to produce very interesting answers. To delete a term from the query, select the appropriate delete (Del:) box, using the upward/downward arrows on the keyboard to reach the box and then pressing the 'Enter' key.

Try now to delete the terms 'yellow' and 'hat', then return to the previous screen. Select 'new Free text query' and put in the query again, namely 'hat yellow orchid stem'.

You now have a query set up and wish to look at the best answer. Do this by pressing the letter 'g' for 'Go to first item'.

Since this is a tiny database, after a second or less, while the computer finds the best answers to the query, the first answer will be shown on the screen as follows:

(computer screen)

colour photograph of Naga objects from various sources: Hat of cane-work with two [missing] black feathers and a large boar's tusk. A chaplet of long pig's bristles encircles the hat interspersed with sharp bristles dyed red. An ornament of plaited yellow cane or orchid-stem is in front attached round the 'chaplet' cane foundation. artefact. Phom. Size:12cm (height of cap)

Production: Phom;
Production: (4:218)
Collection: J.H.Hutton
Acquisition: gift; Pitt Rivers Museum; 1919; (Hutton I.183)
(description derived from original source material unless square brackets or otherwise stated)

Help * Return * Show * Terms * Mark *

This is a fairly full record, describing an artefact in a museum. The different parts of the record describe the following things:

colour photographs... - this is the title, describing what the source or medium is, it might be a short book title, description of set of photographs, a film, a book etc.

Hat of cane-work... - this is in a different colour/tone, to indicate that it is the 'caption' or short description of an artefact, photograph, piece of text or film.

Production: Phom. This is the name of the tribe (Phom) who produced the hat.

Size: 12cm. - gives the (longest) measurement; in this case we are told that this is the height of the cap.
Production: (4:218) - this is a production reference number (in this case the reference number for the photograph of the artefact)

Collection: J.H.Hutton - the collector's name

Acquisition: gift; Pitt Rivers Museum, Oxford; 1919; (Hutton I.183) -
this gives the mode of acquisition (gift), who acquired it, the date of acquisition (1919), and the museum reference number.

There is then a comment in brackets to explain how the record was made.

At the bottom of the screen are various choices, which will be explained shortly.

This is a fairly complex record; obviously films and field photographs and paragraphs of texts will have a simpler description.

We now have a record describing a particular artefact in a museum. Now try pressing 's' or the 'Show' box at the bottom of the screen, and the following will appear, either in reality or in a simulation.

(computer/television screen)

Frame 642 (black and white) (T)ext on/off

[picture of hat - if videodisc attached]

Help * Return * Mark * Text on/off *

If you press 't' once it will turn off all the text on the screen (which is useful if a picture is being shown). Press 't' again to turn the line on again. Try this out.

(A choice 'Next' will appear on the bottom line if there are two or more images associated with a record. For instance, 'Next' will appear if there are two photographs of the object, a front view and side view.)

Now return to the 'base page' by typing the letter r, that is the page which allows you to make a free text query and type:

new free text query> fish poison daughter burial Earls Colne shaman

Now ' Go to the first item' and you will be shown a piece from a burial register for Earls Colne. You have been shown this first because it has four of the words, 'daughter burial Earls Colne' in it. Then type the letter i (or select the 'new Item' box) and you will be taken to another burial. Keep selecting i (new item) until you come to a record about a film. This has two of the words you asked for, fish poison, in it and hence has been shown next. Go back to the previous record (of a burial) by pressing p (or the 'Prev' box at the bottom of the screen), and then press 'n' for next, and you will be back with the film record. Displayed at the bottom of it will be all the available choices, as follows:

(computer screen)

16mm colour film taken by Ursula Graham Bower between 1940 and 1944: fish poisoning expedition at river near Hangrum. films. films. Zemi.
Production: Ursula Graham Bower; 11.1940
Acquisition: Pitt Rivers Museum Archive, Oxford.
* long shots of procession walking to river.
* close-up of people carrying vegetation.
* beating poison into river.
* man beating fibre.
* little boys searching for fish. * little boys searching for fish.
* little boys searching for fish.
* men beating fibre on rocks.
* men scrambling up river looking for fish.
* crowd of men swimming down river looking for fish.
(black and white photographs taken by Ursula Graham Bower * *)

Help* Return* Show* First* Prev* Next* new Item* Terms* Mark*

It is worth reminding ourselves first of what the record itself describes:

16mm colour film.... - this describes the source of the image we will be looking at.

fish poisoning expedition... - this is the short description, giving an over-all caption for the film sequence

Zemi - this is the ethnic group, the Zemi Nagas.

Production: Ursula Graham Bower... This gives the name of the producer (film-maker) and the date (November 194O) when the film was shot.

In this case there is a general sequence on fishing , split into ten separate 'shots', which may be stills taken from moving film, or a sequence of moving film. Each of these has a sub-caption.

If a list of boxes appears like this, they can be selected in the usual way by moving to the box you want to see and pressing on it. This allows you to move through photograph sets, for instance.

Try looking at one of the moving sequences by selecting one of the boxes.

If you have chosen a set of 'stills' from a film, there will be a 'Next' box. Press 'n' to see the next still.

If you have chosen some moving film, there will be a 'show' box at the bottom of the screen and you will now see the first frame of a sequence of moving film, or a message indicating that this is a simulation. Press 'S' or the 'Show' box, and you will be shown the film in motion. If it ends and you want to see it again, select 'g' for Go from start. Type 't' at any time to turn the text overlaying the picture on and off.

Press 'S' again, for Stop while the film is running and a set of controls will appear at the bottom of the screen. These do the following:

Go : sets the film playing

Backwards/Forwards: this is a 'toggle' which plays the film through backwards or forwards
Speed 1/2 : another toggle, which sets the speed to slow motion (1) or normal speed (2), if you press the numbers 1 or 2.

Prev: takes you one frame back

Next: takes you one frame forward

Mark/unMark: marks the still frames (and unMarks them)

Text on/off: another toggle to turn off the overlaying text.

Try each of these controls in turn if you have the computer attached to a videodisc, or imagine how it would look if you do not. Then return to the record, that is the description of the film sequence, by going 'r' twice.

You are now in a record which is part-way through a set of records which have been found in answer to your query. Now try the 'f' (first), 'p'(previous) 'n' (next) and 'i' (new Item) choices a few times to get the idea of how you can move backwards and forwards through the records. Basically, 'first' takes you to the first record you saw, 'previous' takes you to the record you have just seen, 'next' takes you to the next record (which you have already seen), and 'new item' will take you to the first item which you have not already seen. If there are no more 'new items', this choice will disappear from the screen.

Using these controls, find the film record again. You will notice that there are two boxes at the bottom within a bracket. These are cross-references to still photographs which were taken of the same fish poisoning. Select one of these boxes and you will be taken to a (simulation of) the relevant photograph. Press the letter r to return to the film record.

Now find your way to the record which clearly describes a fieldwork notebook entry about were-tigers and shamans. Here you will see the same structure on the screen; the source, a short caption, some keywords, the longer text, and some further keyword fields. This is how a text record looks if it is put in as an ordinary record.

If, however, you have a very long text (such as a book or long diary) which you want to be able to read through page by page, as well as going to separate pages with specific searches, then the text will be put into the computer in a different way, as explained in exercise 5 of Tutorial 1. Here we will just show you how such a record would appear, though you will not be able to try this out unless you have the full Naga database.

In answer to a query on 'leg tattooing of girls', for instance, you might get the following screen:

(computer screen)

manuscript - Christoph von Furer-Haimendorf, Naga diary 3:
leg tattoos of girls. diaries.Shankok. Konyak. Wakching.

23.12.1936.
Production: Furer-Haimendorf; 28.11.1936-11.2.1937.
(translated from german by Dr Ruth Barnes)
Acquisition: School of Oriental and African Studies Library, London

Help * Return * Show * new Item * Terms * Mark *

This is a record which again contains different bits of information. The medium or source is a manuscript diary; the short caption describes the contents of a paragraph of the diary; then follows a name (Shankok), ethnic group (Konyak), and place (Wakching), as well as a date (23 December 1936), to which the paragraph of text refers. The writer of the diary, and the covering dates of that particular diary, are then given under 'Production'.

You are given at the bottom of the screen the option to select 's' for 'Show'. If you did this you would see that instead of going to an image, you are taken to a paragraph of text. In this case, this would look as follows:

(computer screen)

leg tattoos of girls. 23.12.1936. Wakching 23/12/1936

Again storm and rain were beating against the bungalow at night. The morning was heavily overcast but occasionally the sun came out. I talked to Shankok's sister who was coincidentally near the bungalow, into letting me copy her tattoos and she proved to be much less shy about it than I had expected. However Shankok was there the entire time as well. He now wears with pride the shell discs of the head-hunter over his ears and the boars' tusks around his neck. The major part of the girls leg tattoos is already done when they are about eight years old. (112) Slightly later the same lines are again tattooed and at the time of their marriage the tattoos are completed with lines across the knees.

Help * Return * Prev * Next * Mark *

This is the textual record, a paragraph from a diary in this case. It has the short caption in a different colour/tone at the top, with the date and place of writing. The number, if any, in brackets, indicates a new page in the original diary.

In this case the entry is fairly short. When it continues for more than one screen, a choice box 'On to next page' will appear at the bottom of the screen. You could select 'O' for 'On to next page' to see the rest of the text. If you did this, a new box 'Back to previous page' will allow you to return to this page, by pressing 'b' for 'Back to previous page'.

Now that you are within a set of paragraphs, it is possible to read through them by using the n (next) and p (previous) letters or boxes at the bottom of the screen. It is thus possible to read through the whole of a diary or set of letters or book in this way, paragraph by paragraph.

When you now return (type 'r') to the record from any page of the text, you would see that you have returned to the record through which you first entered the diary.

In order to look at the next possibility, return to the 'base page' and type the following:

new free text query> colour of hair

Then 'Go to the first item' and you will be shown a diary entry describing some physical anthropology. In the middle of the record you will see a flashing box. Press the 'Enter' key and you will be shown a simulation of a black and white photograph.

This is the way cross-references, which also appear in ordinary records as boxes within brackets, are dealt with. When you type 'r' for return after looking at the item cross-referred to, you will be taken back to the record or page of text which contained the cross-reference.

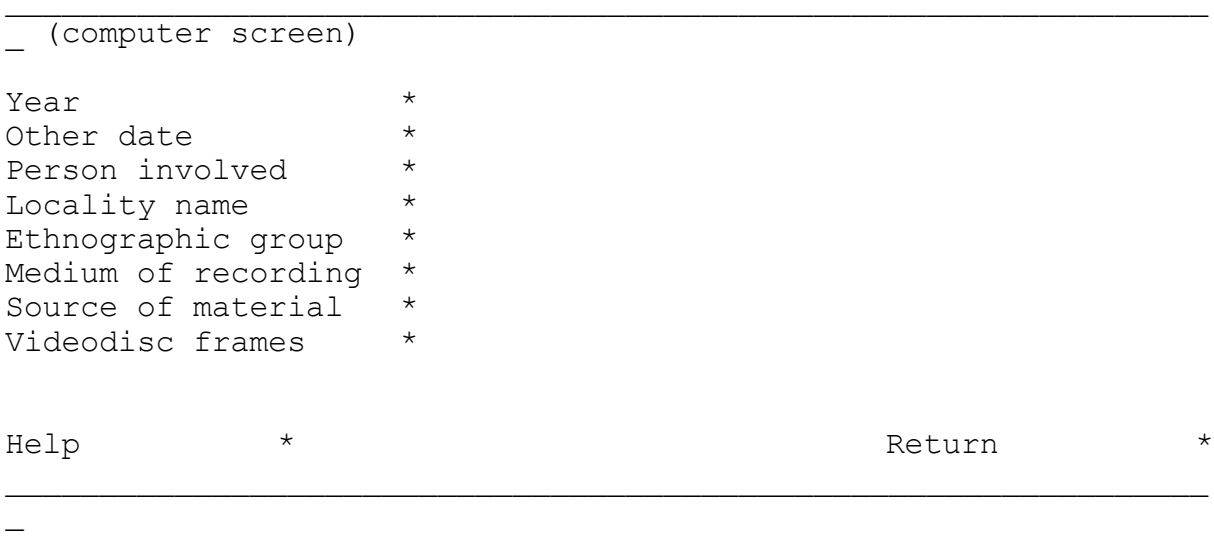
One other type of material is worth exploring briefly here. This is sound material, that is music and speech. Try typing a free text query with the words 'wax recording'. You will be taken to an item which is clearly a sound recording. Select the 'Show' box at the bottom of the screen and you will then be taken to a blank screen with another show box on it. This is paused at the start of the sound. If you select 'Go from start', you will hear the sound, or told that it is playing (in simulation). If the sound is playing and you want to interrupt it in the middle, press 'r' or select the return box.

You have now completed exercise two and are in a position to make free text queries.

EXERCISE 3

Structured and combined searching.

In order to look at the other kind of searching, 'Structured Queries', press 'r' until you return to the initial menu choice and then select 's' for Structured query. You will then see a screen as follows:



Whereas in 'Free text' searches, you could put in any words that came to mind, in 'Structured' queries, you will select a particular field, and then be presented with all the words by which that field has been indexed.

The fields which are indexed in this way, and the choices they allow, are as follows.

'Year' - to select all the records within a year

'Other date' - to select the records by day or month

'Person involved' - to select the records by person's name

'Locality name' - to select the records by place names

'Ethnographic group' - to select the records by tribe or ethnic groups

'Medium of recording' - to select the records by medium of recording (e.g. photograph, film, artefact).

'Source of material' - to select by museum, archive, library

Videodisc frames - to select a record describing a frame number.

Let us first look at how you select by year. Select the 'Year' choice, and you will be taken to several ranges of years.

Choose the one you would like, and you will be taken to blocks of years. Again press on the one you would like, and the year will appear on the right hand part of the screen.

If you would like the records for more than one year, select those you would like, and they will appear in the form, for instance, '1932 or 1933 or 1934'. If you have selected a year you do not want, press on that year box again and it will be deleted.

Try selecting a few years and then deleting some of them (for instance 1936 or 1937), ending up with one or two years. Then return back to the screen which has 'Go to first item'. Now press 'g', and you will soon be shown the first record of the year you selected. A message will also appear at the top of the screen, stating how many records for that year have been found. It might say, for instance '127 retrieved'. If there are more than 1000 records retrieved, you will be told, for instance, '1000 out of 1756 records retrieved'.

If, at any point in a structured query you want to cancel the whole query, you will see a 'Delete' box at the bottom of the initial page which gives you choice of person, ethnic group etc. Select this, or press 'd' and the old query will be deleted.

Go back now to the structured query initial page, delete the old query, and select 'o' for other date. You will now be presented with a request to enter a date in the form yyyy or yyyy/mm or yyyy/mm/dd.
This means that you could enter the date as any of the following:

1936
1936/08
1936/08/26

the last of these meaning the twenty-sixth of August 1936. On typing Enter (carriage return), you will be taken to a list of dates. You can then select one or more dates from this list.

Try selecting a couple of days of the year, and you will be shown all the records made on those days.

Now try the other types of structured queries, people, ethnographic groups, medium, source of material. In each case you will be asked to supply a word. If you supply nothing, in other words type 'Enter' without any word, you will be taken to the start of a list. This would, for instance, be a way of seeing what are the names of the people or places in the material. Or you can type in one or two letters, 'a' or 'ab' or

'abrah' or 'abrahams'. These will take you to slightly different places in the list in each case. Try this out.

You can go on down and up this list by using the 'c' or continued, and 'b' (back) keys or boxes at the bottom of the list. You can select names, places, etc. as you need. The query will then again be built up on the right hand part of the screen.

You can combine different sets of terms, which are joined by 'and', each set having within it 'or'. For instance, try the following. You want to look at all the photographs taken of the village of Chepoketami by Stonor. So you select 'photographs' under medium, 'Chepoketami' under locality name, and 'Stonor' under person. Then run the query and see the result.

There are two further refinements. You may be interested to read about them, but neither of them will work with the small sample database. They both refer to possibilities which you could explore later, and currently exist on a particular application (the Naga videodisc and database). If you do not want to read about them at present, go on to the sixth paragraph before the end of Exercise 3, starting 'Now that you have seen...'

Select the 'Locality name' list at some point and have a look at it. If you have the full Naga database you will see that some of the places are just a single word. In others, there is an = sign, followed by another word. This reflects the fact that places are often spelt in very different ways. When there is an = sign, this indicates that what comes after it is the standard name of the place. If you just want to find the variant, select the name before the =. You will then be shown the map record with all the variants on it. If you want all the records, with all the variant spellings, which are indexed by the standard form, then go to the word that appears after the = and select that.

When you do select a standard place name in a structured query, the first record you will be taken to will be a locality record. This has the following form:

(computer screen)

map * map record *

standard name = all the synonyms

You will be given all the variants of the standard name.

If you select 'map', you will be taken straight to the map which has the standard name you have selected on it. Try that. Then return to the previous screen and try 'map record'. You will now be taken to the following screen.

(computer screen)

map *

covering map *
adjacent maps:
NW * N * NE *
W * E *
SW * S * SE *

Help * Return *

If you select 'map', you will be taken to the map which has the name of the place which brought you to this map record. If you want to go up to a higher level map, then select 'covering map', which shows you the region within which this particular map is to be found (which will contain one or two names which will give you an idea of where the sub-map is located). Typing 'r' for return will take you back again to the lower level map record.

If you want to move to adjacent maps, this is done as follows. Imagine that you are currently in the middle of the set of maps, with other maps to the North (N), North-East (NE) etc. If you want to see the map adjacent to the East of your current map, move to the box marked E and select it. This then becomes the current map. If you select it, then you will be shown that map.

Now, if you want to return to your original location, you would imagine yourself in the middle of the set of maps and obviously your earlier map would be to the west, so you would choose W or west. Alternatively, if you type 'r' for return, you will return to the earlier map record. Try this out.

Now that you have seen how to set up free text and structured queries, select the 'b' (Both free text and structured query) choice in the first screen. This allows you to run a free text query within a structured one.

Supposing, for instance, that you wanted to see all the artefacts in a particular museum which showed a particular

coloured hat.

You would select 'both free text and structured query', then select 's' for structured query. Then choose medium and within this 'artefact'. Then select 'Source of material' and select 'Pitt Rivers Museum'. Then return to the screen which allows you to select 'Free text query' and select that and type in 'yellow hat'. Then 'Go to first record'.

Using the records in Appendix J, devise for yourself a simple
'both free text and structured' query, and see if you can find the record(s) you want.

You have now learnt how to do four different kind of searches; by hierarchical contents, by free text searching, by structured searching, and by combined free text and structured searching.

EXERCISE 4

Altering the retrieval style.

The normal retrieval style is by record. That is to say, the answers to a query will be presented as a set of records. For some purposes it is quicker to retrieve information in other forms. . If you type 'A' for 'Alter retrieval style' when that is presented, you will be presented with a screen:

(computer screen)

Data retrieval	*
Caption retrieval	*
Help	*
Return	*

Select 'Data retrieval' and then try a query. You will now be taken to one of three things.

If you are taken to a picture, either a still image or a start of a film, you can look at this. If there is a 'next' box on the menu bar, then you can go straight to another image or piece of film in a sequence. If there is not such a choice, then selecting 'i' for 'new Item' will take you to the next piece of data retrieved by the query. If you want to see the record which describes the image on the screen, type 'r' for return. If you want to go back to the image again, select 's' for show.

You may, alternatively, be taken to a piece of text, that is the paragraph of a book or diary. Here you can again go back to the record that describes this paragraph with 'r', or read through further paragraphs of the same source with 'n' for next and 'p' for previous.

Thirdly, you may be taken to a record which contains two or more illuminated boxes. For instance, a series of film sequences may be presented. Choose one of these in the normal way by pressing the 'Enter' key. You will then be shown the image. To see the next image or film in this record, type 'r' and repeat on another box.

Note that if you return from the data to the record which indexes the data, you have moved up a level in the system. If you select 'i' for new Item at this level, you will be shown the next record. It is as if you had returned to 'record retrieval mode'. To go back down to the level of data retrieval you need to go down to the particular text or image with 's' for show.

Note also, that if you mark 'data' (images or text), you will be asked for captions. Nor will you be able to 'expand' such items that you have captioned, since they have no indexing terms associated with them. If you want to mark them, you will need to go up to the level of the records associated with the data, and mark those records.

Now select go back up to the base page and select 'Alter retrieval mode' and you will be given the choice of 'Caption retrieval' and 'Full record retrieval'. Select 'Caption retrieval' and a list of short captions is presented, each of which has a box opposite it. Moving up and down this, and pressing 'Continued', or 'Back' if necessary, you can select a caption.

When you do this, you are taken to a particular record, which will take you to an image or text. When you return from this record, you will find yourself in the caption list again. If you re-select 'Alter retrieval style', you go back into selection by Record. Try making queries using the caption mode.

EXERCISE 5

Marking the records and examining or saving them.

You will have noticed on a number of screens the choice 'Mark'. Selecting this allows you to mark records. Let us try this.

Enter free text mode in the 'full record retrieval' style

and find a record. Now type 'm' or select the mark record box. You will now see 'Item marked as relevant' at the top of the screen. If you want to change your mind, select the same box, which is now called 'unMark record', and the item will be unmarked again. Try marking and unmarking the record. Mark three or four records like this and then return to the previous (base) screen.

This will now have a choice called 'Keep the marked record', select the box or type 'k' and you will be asked for a caption. Think of a word or a few words which will remind you what the items which you marked were about, for instance 'fishing' or 'fishing with nets'. Type the word(s) you have chosen and enter this caption with an 'Enter'. You will now see that the label has changed to 'Inspect the Marked items'. Select this with the appropriate box or letter 'm', and you will be shown a list. For example you might see:

```
fishing with nets
     Del: *   See *    a fishing expedition with nets
     Del: *   See *    some nets used in fishing
     Del: *   See *    nets which are used by women fishing
```

and so on.

This gives a list of the shortened captions of the records you have marked. If you want to delete one of the records, you can do so by selecting the delete box, if you want to see the full record, press the 'See' box. On typing 'r' you will come back to this list. The list might have several files of items that have been marked during the current session. Try to look at and delete one or two of the records you have marked.

You have now marked some records which have a caption. If you go to a Videodisc image or page of text, using 'show' or a box, then there will be a 'mark' choice. When you press mark in these cases, you will be asked to supply a caption for the image. Thus you can mark a photograph, a piece of moving film, a still from some moving film etc. Try to find some moving film and supply captions in this way. Then return, keep the marked items, supply an over-all caption and look at them. You will now have two sets of items, one a set of records with the captions highlighted, one a set of images or texts with your supplied captions in ordinary letters.

There are two uses for these files of marked items. One is to use them as the basis for building up a file which can be saved after you leave the database. This file can then be edited and re-entered in the database.

If you have the full CDS 2000 system (but not DISCAT on its

own), you can save this set of marks to a permanent file as follows.

When you have marked a set of documents, return to the page which allows you to 'keep' the marked file. 'Keep' the marked file, giving it a name, and then you will have the choice of 'Inspecting the marked file'. Select this, and you will be shown a list of marked items, and on the bottom line there is a chance to choose 'output' and 'input'. If you select 'output', you will be asked for a name. Choose a name and enter it, and then the set of marked items will be written to a permanent file of this name, with the extension .mks in whatever directory you entered by.

When a marked file is 'output' in this way, it is deleted from the database. But if you want to retrieve it, select 'input' at the bottom of the screen, or when given a choice of 'Input marked file' and specify the name of the file you want to retrieve.

When you leave the database, but while you are still in 'muscat', you can print out the marked file by typing as follows:

c-getmrecs filename.mks to filename ('filename' being the name you gave)
(or, if you have a DA system, then use 'c-getdiscm' instead of c-getmrecs)

Then type:
c-print filename to filename1
You will then have an ordinary text file, called filename1.txt, which can be printed out with a word-processor.

EXERCISE 6

Marking records and expanding the query.

The other use of the marking system is a powerful extension of the searching system and we will explore it now.

Go to a particular record (using a short free text query) and press 't' for terms. This will produce a list of the terms by which that particular document is indexed in the database. Against each term is a small box with 'Add' against it. Try adding one or two terms. You will be given a message at the top each time stating that the term has been added to the query. If you return to the previous screen and inspect the free text query, you will find that it now has these new terms added automatically to it. Try doing this with a record.

Now do another query and mark three or four records with 'm' for mark. Return to the 'base page' and 'keep' and 'inspect' the marked list. You will now see a choice 'Expand'. Select this box, or type the letter 'e'. After a short while the computer will produce a list of words, each with 'Add' against it. These are all the terms by which the three or four records you marked are indexed, sorted into an order. The order is one which tries to show the most useful (that is statistically most linked) terms first.

Thus if you had three records, all with the word 'house' in them, two with 'roof' and one with 'thatch', you would get in order house, roof, thatch. But there may be several words which appear in all three records and are hence all equally well correlated. These are then placed in order of their frequency of appearance in the whole database. Thus, if 'house' appears in all three records, but is a very common term, whereas 'blue' also appears in all of the three records, but is a very uncommon term, 'blue' will come before 'house'.

Try adding a few of the more highly associated terms and re-running the query. You will begin to get different results, which can then be marked again, and further terms added. In this way, your intuitions about associations and the computers discovery of statistical associations will be working together. You will mark the records which seem good answers to what you were really looking for, and the computer will suggest other keywords and associated words which improve the query.

You should now be able to explore the materials by yourself. Remember that there are 'Help' screens for most of the choices and you can always return to the start with repeated 'R' keys if you get lost. To exit from the system, type x. (If you are then presented with a muscat> prompt, type 'stop' followed by a carriage return (enter) key.)