Relational multimedia databases.

Sylivano Chiluli NongaKitinya,

Relational Multimedia Databases

Sylivano Chiluli Nonga Kitinya

This thesis is submitted in fulfillment of the requirements

for the degree of Doctor of Philosophy of the

University of Durham

School of Engineering & Applied Science

Department of Computer Science **1987**

ABSTRACT

Relational Multimedia Databases

Sylivano Chiluli Nonga Kitinya

This thesis is concerned with the design and implementation of a Relational Multimedia Database System, in short RMDBS. RMDBS is designed to efficiently use storage space and manipulate various kinds of data; attribute data, bit-mapped pictures, and programs in binary code.

RMDBS is an integrated system which enables the user to manage and control operations on the different forms of data in a user friendly manner. This means that even non-experienced users can work with the system.

The work described in this thesis is novel in that a true multimedia database has been implemented within the framework of a traditional relational DBMS. Previous work in this area has concentrated either in building data base management systems for storing picture-based data or multimedia databases which are not true data base management systems.

RMDBS is implemented using the Revelation data base management system.

ACKNOWLEDGEMENT

I would like to express my sincere thanks to my supervisor, Mr. J. S. Roper for the immense help and encouragement during the course of this project. I also acknowledge many stimulating discussions with Jim Cottrell, the departmental software technician.

The Malcolm MacDonald Studentship provided the funding for the project. Many thanks to them.

Finally I am indebted to Rev. David W. Jones and Dr. John Welford and their families for the help and company they gave me.

COPYRIGHT NOTICE

CONTENTS

Abstract i

Acknowledgement ii

Copyright notice iii

Contents iv

Declaration x

Chapter 1

Overview

1.1 Scope of Thesis 1

1.2 State of the Art 1

1.3 Overview of Thesis 2

Chapter 2

Data Base Management Systems

2.1 Evolution of Data Bases 4

2.2 General Structure of Data Bases 4

2.3 Relational Data Bases 8

2.4 Network Data Bases 10

2.5 Hierarchical Data Bases 11

2.6 Comparison of Approaches 13

2.7 The Revelation DBMS 14

Chapter 3

Computer Graphics

3.1 General 18

3.2 Raster or Bit-Maps 19

3.3 Drawing Text & Lines by Dots(Pixels) 20

3.4 Graphics Packages 22

3.5 Data Structures for Computer Graphics 23

3.6 Computer Graphics Input Devices 23

3.7 Computer Graphics Output Devices 25

3.8 The CORE Graphics System 27

Chapter 4

User Interfaces to Data Bases

4.1 Why User Interfaces? 28

4.2 Command Language Driven Systems 29

4.2 Menu Driven Systems 30

4.4 Forms Driven Systems 33

4.5 Graphics Based Interfaces 36

4.5.1 The CUPID System 36

4.5.2 Query By Example 38

4.5.3 The Ski System 39

4.5.4 The Spatial Data Management System 40

4.6 Summary 42

Chapter 5

Storing Picture-Based Information in Relational Databases

5.1 Introduction 44

5.2 The "Picture-Building" System 44

5.3 A Relational Schema with Colour & Texture 46

5.4 Query-By-Pictorial-Example 50

5.5 The Image Analysis and Image Database Management System 53

5.6 Other Systems 54

5.6.1 GEO-QUEL 54

5.6.2 The Geographic Information Systems Project 54

5.6.3 Query-By-Structure-Example 54

5.7 Conclusion 55

Chapter 6

Multimedia Databases

6.1 Introduction 56

6.2 Document Processing in a Relational DBMS 57

6.3 The Multimedia Office Filing System 57

6.4 A Multimedia Information System for an Office Environment 59

6.5 The MINOS System 60

6.6 The Muse System 61

6.7 Conclusion 63

Chapter 7

Generation & Manipulation of the Pictures

7.1 Picture Processing for Printing 65

7.2 Data Used and Its Sources 67

7.2.1 Digitised Maps 67

7.2.2 Census Data 68

7.3 The Display Processor Used 68

7.4 Picture Generation 69

7.4.1 From Pascal Programs 69

7.4.2 With the Smart Spreadsheet 70

7.4.3 By Microsoft Windows PAINT program 71

7.4.4 Comments on the Generation Methods 71

7.5 Storing the Pictures in Database 72

7.6 Storing Binary Programs in Database 73

7.7 Picture Manipulation in Database 74

7.8 Performance Comparison 77

7.9 The Results 78

7.10 Observations, Recommendations & Conclusions 83

7.10.1 Observations 83

7.10.2 Recommendations 83

7.10.3 Conclusions 84

Chapter 8

The Relational Multimedia Data Base System

8.1 Specification 86

8.1.1 The Making of RMDBS 86

8.1.2 Need for Picture Generating Capabilities in RMDBS 86

8.1.3 A Demonstration System for RMDBS 87

8.2 System Overview 84

8.3 Relations Used 90

8.3.2 For Screen Pictures 90

8.3.2 For Laser Printable Pictures 91

8.3.3 For Binary Programs 91

8.3.4 For Parameters 92

8.3.5 For Digitised Maps 93

8.3.6 For Census Data 94

8.4 Writing of Boolean Expressions 96

8.5 Description of the Functions 98

8.5.1 Get Help 99

8.5.2 Display Picture 99

8.5.3 Insert Screen Picture 99

8.5.4 Insert a Laser Picture 100

8.5.5 Insert a Program 100

8.5.6 Print a Laser Picture 101

8.5.7 Print a Screen Picture 101

8.5.8 Scale a Picture 102

8.5.9 Overlay two Pictures 102

8.5.10 Run Smart 103

8.5.11 Run Microsoft Windows 109

8.5.12 Draw on a Picture 112

8.5.13 Descend to Main Revelation Menu 113

8.5.14 Delete "least used" Pictures 113

8.5.15 Set/Update Parameters 114

8.5.16 See names of Programs & Pictures 114

8.5.17 Delete Picture or Program 116

8.5.18 Descend to TCL 117

8.6 Conculsion 117

Chapter 9

Conclusions & Future Work

9.1 Conclusions 119

9.2 The User Interface and Its Limitations 121

9.3 Thoughts on Future Work 123

9.3.1 Basic Extensions 123

9.3.2 User Friendliness 125

9.4 Final Conclusion 127

References 128

Appendix 0: Hardware Configuration 136

Appendix 1: The Revelation DBMS 138

Appendix 2: The Smart Spreadsheet 143

Appendix 3: Microsoft Windows 148

Appendix 4: Turbo Pascal 150

Appendix 5: The DEC LNO3 Laser Printer 156

Appendix 6: The RMDBS Main Routine 158

DECLARATION

The material in this thesis has previously not been submitted for any degree in this or any other university.

Chapter 1

Overview

1.1 Scope of Thesis

This thesis is concerned with the design and implementation of a relational multimedia database. It is intended that the system be able to achieve first, optimum use of storage space and second, to be "user friendly". The system is user friendly in the sense that a non-experienced user should be able to work with the system without too many problems.

1.2 State of the Art

Following the wide use and acceptance of data base management systems, several issues have come to dominate their uses:- performance, distributed databases, interfaces, multimedia databases. Relevant to this thesis are the two issues, namely: (i) user interfaces and (ii) storing data of different forms(multimedia) in the database.

The issue of user interfaces is important because of the fact that as databases become more widely used, most users of these databases will be non-experts in the field of computers. This means that interfaces to databases have got to be "user friendly" to enable naive users to have access data in them.

Multimedia databases have come to the forefront because of the following reasons:

(a) In most cases, information is not attribute data only (as early designers of data base management systems planned), it comes in many forms; images, charts, voice, etc. To be able to store and process such information, it needs to be stored and manipulated in multimedia databases.

(b) Following the rapid improvement in storage technology, various types of storage media have been produced and become economic to use (for example optical disks), therefore new

types of database management systems have to be designed to take advantage of such developments.

(c) Processing information in different forms enhances understanding of the information. For example, giving a user information in number form sometimes does not make more sense than if the user is given a chart representing the same data. Adding a map or another plot to the chart can make an even bigger impact!

1.3 Overview of Thesis

This thesis is concerned with the design and implementation of a multimedia database with a user friendly interface.

The central concepts of Data Base Management Systems are discussed in chapter 2. The main vehicle of which the system is implemented is the Relational DBMS, Revelation. Details on the Revelation DBMS can be found in Appendix 1.

Chapter 3 discusses computer graphics. Since images/pictures are used extensively in this thesis, this serves as an introduction to bit-mapped and vector pictures, display processors, and graphics packages.

Chapters 4 to 6 review work done on the integration of graphics information into DBMS.

The issue of user interfaces to databases is discussed in chapter 4, where a survey of different ways of generating user friendly interfaces is made. The exposition also includes several graphics-based user interfaces. This is the first step in integrating graphics information into DBMS.

The evolution from traditional databases to multimedia databases is discussed in chapters 5 and 6. In chapter 5, several systems which store picture-based information (and the way in which they generate the pictures) are discussed. Chapter 6 describes the design and workings of several, fully fledged multimedia databases.

Chapter 7 describes how the pictures are generated and manipulated. The pictures are

generated from several systems; Turbo Pascal, Smart, and Windows (see Appendices 2, 3, and 4 respectively for details on these systems). These picture are then stored and manipulated in the Revelation database. It is also shown how binary programs can be stored and used in the database.

Following the successful demonstration that various types of data can be stored and manipulated in a traditional DBMS (from chapter 7), a relational multimedia database is designed and implemented. This system is described in chapter 8. The specification of the system, its use, and its on-line manual are all described in the chapter. Appendix 0 lists the hardware configuration on which this system is implemented.

Chapter 9 lists conclusions arising from this work. Also included are thoughts on future work.

Chapter 2

Data Base Management Systems

2.1 Evolution of Data Bases

The data base is a natural and inevitable outcome of the progressive development of data processing techniques[1]. This development can be traced through several stages. Initially as we know, the first commercial systems were simple applications using serial files. From this simple genesis more complex systems have come to being, among them are report generators, packages, and management information systems.

The aim was to have an integrated collection of files and programs that could work together and lessen or solve the two fundamental problems in the use of files, namely, the data independence and the data redundancy problems[2,3].

Data independence aims to attain the ideal programs that are independent of the file organisation methods, such that the programs processing the file will still run regardless of changes in the file organisation methods. On the other hand, the data redundancy problem is the problem of ensuring that each item of data appears only once in the master files, because multiple occurrences of a data item wastes storage space and increases the risk of inconsistency in the stored data.

2.2 General Structure of Data Bases

A data base with its software for manipulation and maintenance, taken as a whole is referred to as a Data Base Management System(DBMS). Broad agreement about the organisation of these systems exist, see the ANSI/SPARC[4] agreement, the GUIDE/SHARE[5] and CODASYL[6] reports.

To solve the data independence and data redundancy problems, DBMS are organised in four levels as illustrated in figure 2.1.

A user is defined as anybody who needs to access a data base. The user, therefore, can be an application programmer(writes programs which access the data in the data base) or a terminal user(uses some sort of query language at the terminal to access the data in the data base). Each user is given a work area where data is placed by the DBMS. At the disposal of each user is the Data Manipulation Language(DML), called Data Sublanguage(DSL), which is concerned with the retrieval and storage of data in the data base. The DSL can be embedded in a host language like COBOL or it might be an extension of an already existing programming language.

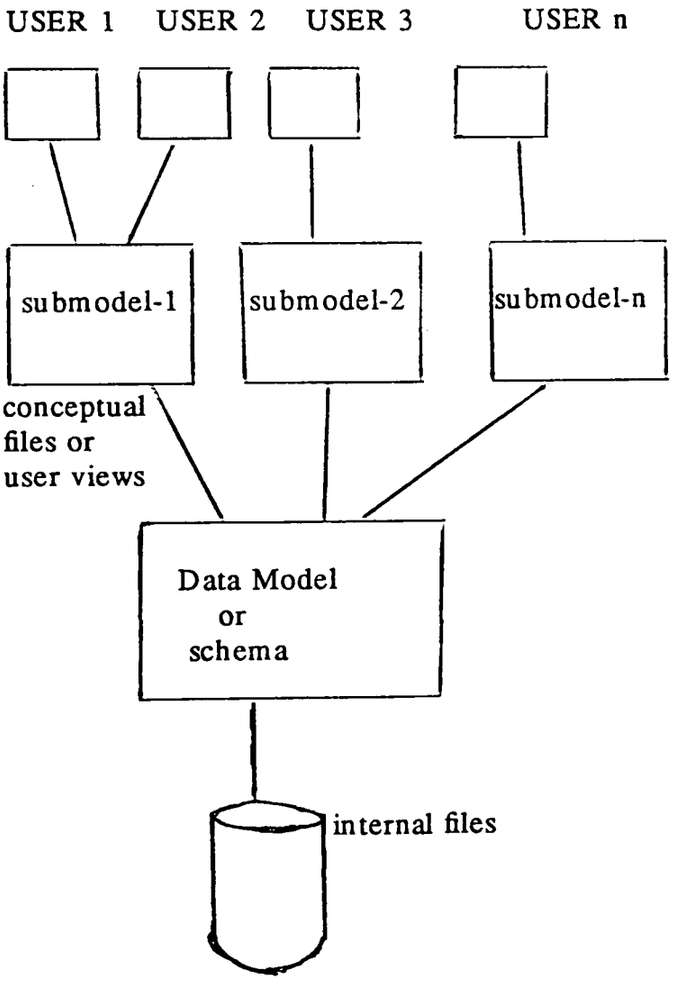


Fig 2.1 Data Base Organisation

The data base itself is physically stored on discs or drums. This is known as the storage structure sometimes known as internal files or physical files. Organisation of records in the

storage structure can be sequential, indexed sequential, random, etc. The user need not know how the records are stored.

In between the physical data base and the user are the Data Model, the DM, or Schema and the Submodels or subschemas.

The Data Model or Schema contains the description of the whole data base. It defines the various record types used and how these records are mapped into the internal files. It is the dictionary of all the data items used.

Each user sees the data base through his or her submodel or subschema. This subschema is called a conceptual file. A conceptual file does not physically exist but exists logically.

It is the responsibility of the Data Base Administrator, the DBA, to define the structure of the internal files, the data model, the submodels, and the mapping functions needed to map conceptual records to their internal files. The DBA uses a Data Description Language, DDL, to accomplish these tasks.

Now, to see how the data independence problem is solved by use of a DBMS, consider USER-i with program X, which manipulates the conceptual file with fields A, B, and C as shown below:

A	B	C

Because the above record is a conceptual record, the internal record may be different, and normally is, from this and can contain other fields. USER-i does not see these other fields. This is the only submodel program X sees. The user also does not know how the physical records are stored in the internal files. Program X manipulates these physical records via mappings through the submodel and the schema. Any change, therefore in the physical structure does not affect program X, so X enjoys data independence.

If another program Y from USER-j needed to use a file with fields A, B, C, and a new

field D. What is only required here is that, the DBA alter the conceptual file to one containing the following:

A	B	C	D

So a new conceptual file is made which USER-j uses. Again program X is not altered.

Data redundancy, due to duplicated data is now under control because programs or users share data through the conceptual files. There is thus no need for each application to have its own private files. In actual fact any duplication, if any, is put there by the DBA after considering other factors like operating efficiency.

Another advantage coming from the use of data bases is control of the data. Through the DBA, data is placed under one control. Since an organisation should be able to control its valuable assets, through the DBA the organisation can exercise control on its information. Another advantage is that the DBA can enforce standards on data item formats, names, etc a thing not possible without the use of the DBMS.

Placing the company's data under one control brings and solves other problems, these are:

(i) Security

The DBA must ensure that data can be accessed through the proper channels only. Through the use of conceptual files and the schema, the DBA can define checks users must be subjected to before given access to a piece of data.

(ii) Integrity

This is the problem of ensuring that the data in the data base contains only accurate data. This means the DBA must ensure that data is updated correctly, and if several entries of the same data exist, all copies are correct. At the same time, recovery procedures to restore the data base to a correct state must be at hand in case there is a hardware failure.

Three broad categories of DBMS exist depending on how the data is organised, these are: Hierarchical, Network, and Relational[7]. The main features of each of these is briefly described in the following three sections.

2.3 Relational Data Bases

The relational data base model was introduced by Codd in his acclaimed paper[8]. More details on the organisation of data in this model can also be found in[2, 9, 10].

The main characteristic of this model is that files are stored as relations.
The term relation in mathematics means the following: Given sets D1, D2, D3, ..., Dn a relation R is a set of n-tuples each of which has its first element in D1, the second in D2, and so on. The sets D1, D2, D3, ..., Dn need not be distinct. The sets Di are known as domains of R. The value of n is called the degree of R. The number of tuples in R is called its cardinality.

The definition of a relation further means the following:

i) no two rows are identical

ii) the ordering of rows is immaterial

iii) the ordering of columns is significant

Another popular way of representing a relation is a table. In the tabular representation, it is customary to name the table and to name each column. The columns of the tables are called attributes.

A column or a set of columns whose values uniquely identify a row in a relation is called a candidate key or just a key. Since a relation can have more than one key, one of them is designated as the primary key.

Figure 2.2 is an example of table relation with five columns or fields. The first one, NAME, is the key. The name of the relation is COUNTIES.

COUNTIES

NAME	X	Y	POP	SIZE
SOUTH GLAMORGAN	30815	17457	377	4183
ISLE OF WIGHT	44970	8627	115	3795
KENT	59020	15134	1448	37347
TYNE & WEAR	42788	56222	1136	5424
GREATER LONDON	53133	17970	6609	15792
WILTSHIRE	40478	15740	513	34727
POWYS	30372	26898	108	50699
STRATHCLYDE	21725	66717	2375	137877
BERKSHIRE	46434	17280	671	12557
DERBYSHIRE	42615	35981	902	26264
GLOUCESTERSHIRE	39190	21443	493	26409

Fig 2.2 Example of a Relation

If relations are to be used in a DBMS without creating update or deletion problems, they have to be normalised. Normalisation is explained by Codd in[11] and more rigorously in[12]. What normalisation achieves is to make the relations more regular in the sense that each represents only one fact. Unnormalised relations exhibit update and deletion anomalies because of the functional dependencies that may exist between attributes. Normalised relations also reduce the need for restructuring the relations when new types of data are introduced. This increases the life span of application programs.

There are three types of normal forms; first normal form, second normal form, and third normal form. Any relation in first normal form(is a flat table, without any repeating fields) can be manipulated by relational DBMS, but for optimal performance relations must be in third normal form.

In most cases normalisation for second and third normal forms is treated in a non-algorithmic way, requiring a human being to recognise a pattern. When a pattern is recognised, a file is split into smaller files. Pattern checking for these smaller files is then repeated and splitting performed if patterns are recognised. This method is fraught with problems because errors can be made if the human being does not see the correct patterns. However, nowadays algorithmic methods are available for decomposing files into third normal form. For example, Salzberg in[13] discusses the Bernstein method.

The DMLs used to manipulate relational data bases are based on relational algebra or relational calculus[14].

Both types of languages are equally powerful[15], and any user query which can be expressed in the form of first order predicate calculus can be handled. Relational calculus languages are non-procedural. They are non-procedural in the sense that a user specifies his request to the DBMS by defining the property of the information he wants instead of specifying the instructions(procedure) to obtain the information. Expressions in these languages are first order predicate calculus expressions. Relational algebra languages are procedural languages and instructions in these languages consist of simple instructions that are similar to operations (union, intersection, difference, etc) of mathematical set theory.

2.4 Network Data Bases

The concept of network data bases was first expounded by the Data Base Task Group(DBTG)[6]. Details on the organisation of these data bases, in a tutorial fashion can be found in[16, 2, 10].

The basic data structure of network data bases is the data structure diagram as introduced by Bachman[17]. This structure notation uses two fundamental components; a rectangle denoting a record type and an arrow or pointer. The arrow connects two record types. The record type located at the tail is called the owner-record and the one at the head is called member-type. This owner-member named directed arrow is called a set. Several set types exist: one-to-one, many-to-one, one-to-many, many-to-many, and recursive, that is, a record to itself. Sets are used to permit data base traversal in response to a user query.

Figure 2.3 is an example of a network of five record types with six sets.

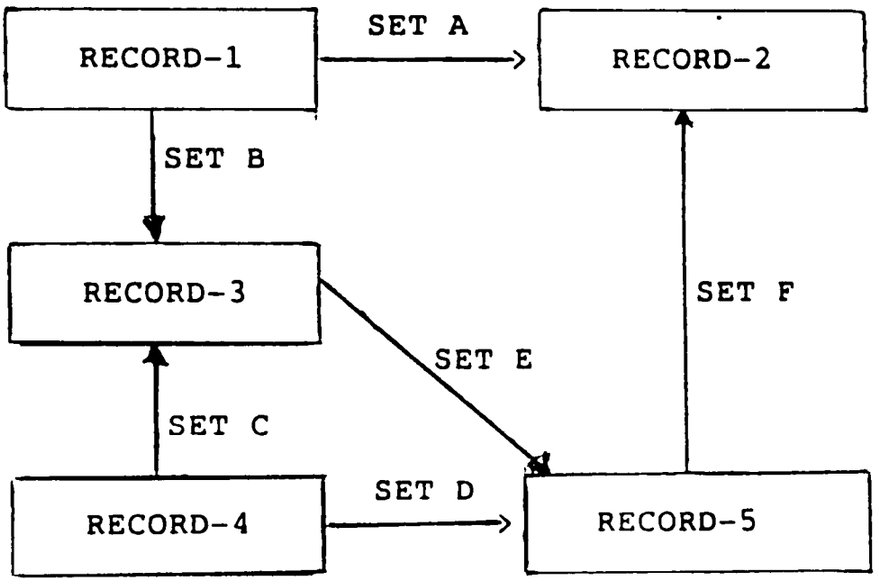


Fig 2.3 Example of Network of Record Types

The processing language, the DML, in a network structured data base is a procedural language and in most cases it is an extension of COBOL or PL/1. The processing is termed procedural in the sense that in every set occurrence:

i) given an owner record, member records can be processed.

ii) given a member record, the owner record can be processed.

iii) given a member record, other member records can be processed.

To access records in a network data base, the user must navigate a route through the records. This implies that one needs to know how the data base is organised for proper navigation.

As with the DML, the DDL which is used to describe the schema, that is, the records and their relationships, is usually an extension of COBOL or PL/1.

2.5 Hierarchical Data Bases

The basic data model in a hierarchical data base is the hierarchical relationship between records types. The driving idea behind this organisation is: because hierarchies are familiar in nature and in human society, and since data bases portray and represent information about us,

it seems logical to design them hierarchically[18]. The main document describing hierarchical data bases is[5], and further details can also be found in[2, 10].

Record type relationships can be viewed as 1:N. The 1:N relationships have directions represented as arcs in a graph[17]. The data base, therefore, corresponds to a forest of trees whose nodes are the records. Figure 2.4 is an example of hierarchical relationship between two record types, COURSES and STUDENTS. Each COURSES record is related to many STUDENTS records, these are the students in the course. Figure 2.4(a) shows the relationship between the two record types and figure 2.4(b) is an example of the appearance of some data and their relationships.

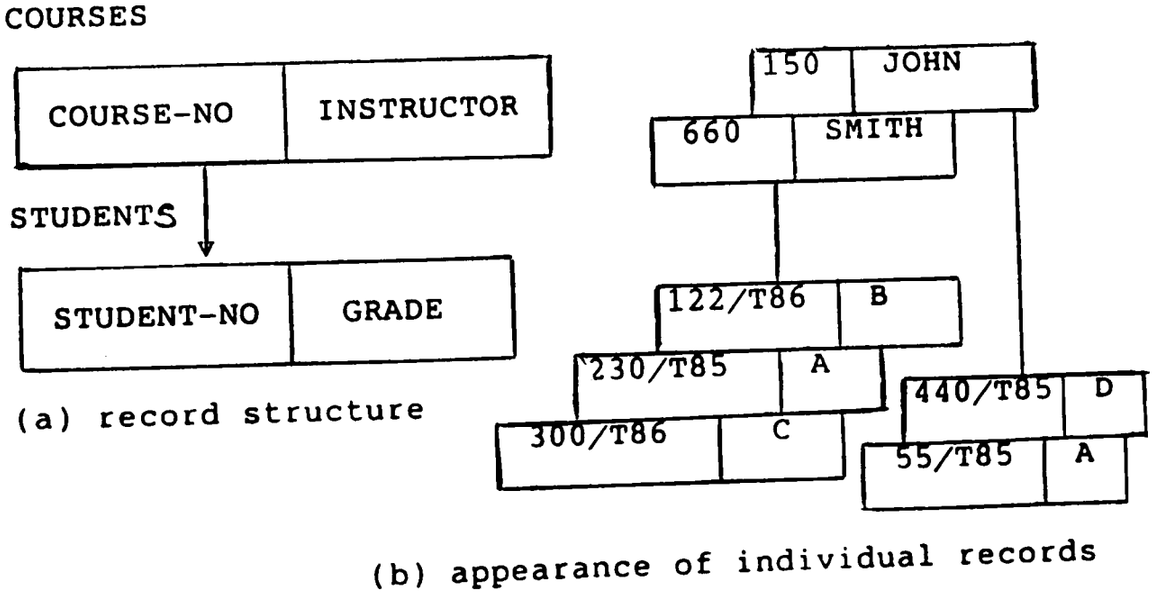


Fig 2.4 Example of Hierarchical Records

A typical hierarchical system is the widely used Information Management System, IMS, developed by IBM.

Similar to network data bases, hierarchical data bases are processed by procedural languages. In particular records are processed in tree traversal form[19]. Again as with network data bases, these languages are normally extensions of COBOL or PL/1.

2.6 Comparison of the Approaches

There has been much debate on the merits and demerits of the three data base approaches[20, 21, 10, 2]. Although there has not been a clear winner, and the situation will continue for a long time[21], relational data bases though coming later, (latest products include INGRES[22, 23] and System R from IBM) have become very popular.

NOTE: The hierarchical DBMS IMS and the DBTG product IDMS have been in wide use for some time.

The main advantage of the relational model is that it is based on a very firm theoretical foundation[8]. Its representation of files as flat tables adds to giving a uniform view of the data base. The sublanguages used to manipulate relational data bases are not procedural. This means that even someone who does not know how the data is organised can still get data from a relational data base. Furthermore, adding a relation to the data base or a new attribute to a relation does not impact existing application programs. According to Tsichritizis and Lochovsky[24], this implies that relational data bases are more data independent than network or hierarchical data bases, and therefore better.

Recently, however, the debate is showing signs to move from the discussion on the merits or demerits of the data base models towards:

(i) standardisation[25, 26, 27, 28], and

(ii) design of generalised data models capable of representing any of the data base models[29, 30].

Standardisation wants to classify data base management system related components into both internal and external components in terms of functions. The aim being to permit buyers of data base management systems to mix and match their software to the same extent as buyers already do with hardware from different vendors[28].

The design of a generalised data base system is based on a canonical data model, that is, one which is potentially capable of supporting user views of all models through local schemas and appropriate data manipulation languages[29]. The design of Prototype of RElational Canonical Interface(PRECI)[30] is a good example. Using such a generalised data model, for

example, a hierarchical or network data base might appear to the users as a relational data base and vice versa.

As mentioned earlier, hierarchical data models are tree structures and network data models are graph structures. Since tree structures are also graph structures, hierarchical data models can be represented by network data models. That is, there exists a one-to-one mapping from the hierarchical data models into the network data models.

In a network data model, the logical structure is completely described by a graph consisting of a collection of record types and a collection of arcs called sets. Each record type corresponds to a relation in a relational data model. An arc that joins two record types can be represented by a relation consisting of two domains. The first domain consists of record keys of record type RECORD-A and the second domain then consists of the corresponding keys of record type RECORD-B. Therefore, if there is an arc joining record n of RECORD-A to record m of RECORD-B then this arc is represented by the tuple (KEY-A-n,KEY-B-m) in the relational data model. Since both record types and arcs(sets) in the network model can be represented as relations, any network model can therefore be represented by a corresponding relational model. That is, there exists a one-to-one mapping from network data models into the relational data models.

As was shown above, hierarchical data models can be represented by network data models and network data models by relational data models. Therefore, the relational data model is capable of representing the logical data structures of both the hierarchical and network approaches. Taking advantage of the generality of the relational approach, this research therefore, uses the relational data model as a basis for data base management systems.

2.7 The Revelation DBMS

Because the research uses the Revelation DBMS to store information, it is important that some features of Revelation are explained. This section, therefore explains the main characteristics of the Revelation DBMS. Details can be obtained in[31,32].

Revelation uses accounts, files, and records to manage data. Accounts are there to separate one type of user or application from another. An account may contain any number of data files. Data maybe held in any account and is accessible from other accounts.

Each record in a Revelation database has a unique label, the key or record id. A record can have only one record id and a record id refers to only one specific record. Record ids can be any combination of fields.

Revelation considers each column in a record as a separate field. Each field can be variable length, that is, the field may contain as many characters of data as is needed. (The only limitation is that Revelation software places a maximum length for records of 65,536 bytes). This is one of the key features of the Revelation system. Revelation also supports multi-valued fields, that is, relations not in normal form.

Revelation files are called "relational files". This simply means that the information in a record could be presented in a tabular form. Although Revelation allows one to use unnormalised relations, in this research all relations used are normalised, except when storing the xy vectors for a map/figure. In this case, multi-valued fields have been used. However, this is a necessary requirement when processing such types of relations, see also[33, 34].

Revelation is called a "dictionary driven" system. When a command is entered, Revelation checks the dictionaries for the meaning of each word before executing the command. There are two types of dictionaries; data dictionary and the VOCabulary file.

Data Dictionary

There is one data dictionary for every file. It is created when the file is created. The data dictionary contains all the information about each and every field. Up to fifteen parameters(name, output conversion, validation criteria, description, etc) can be specified for each field.

The VOCabulary file

The VOCabulary file (VOC file) is considered to be a dictionary because it contains an entry for every word, command and symbol that Revelation understands. Every account has a VOC file. The VOC file can be customised by the user. For example a Swahili user might like to use the word KAMA instead of WITH.

Revelation has the following major components:

(1) R/DESIGN

This is Revelation's 4th generation menu driven applications language. With R/DESIGN one can define file contents, generate programs to enter data into files, produce reports, etc, without descending into programming.

(2) R/LIST

R/LIST is Revelation's query language. Although R/LIST is a powerful propositional calculus based language, it has the limitation that only one relation(file) can be specified in a query. To refer to fields in other files, this must have been anticipated and done in R/DESIGN through the XLATE function.

(3) R/BASIC

This is Revelation's programming language. From within R/BASIC one can include an R/LIST command or perform any DOS operation.

(4) R/LAN

R/LAN(Local Area Network) is the name of the networking version of Revelation. This network uses network operating system semaphores to lock data at record level. These semaphores allow shared files to be used simultaneously by several stations on the network.

(5) R/EDIT & R/TEXT

These are editors; R/EDIT is a line editor, while R/TEXT is a screen editor.

Chapter 3

Computer Graphics

3.1 General

The roots of computer graphics can be found in early attempts to use the Cathode Ray Tube(CRT) as an output device instead of remaining purely part of a computer engineer's tools[35]. However, computer graphics took on a new turn after Sutherland's epoch making work on SKETCHPAD[36]. Since then, computer graphics has grown to become one of the fastest growing areas in computer science[37, 38].

The CRT uses electric fields to generate a finely focused, high speed beam of electrons, and to deflect the beam to various points of the screen surface so as to generate a visible trace. To be used as an output device, the device is constructed with the usual CRT components plus a secondary cathode located between the primary electron gun and the screen, see fig 3.1.

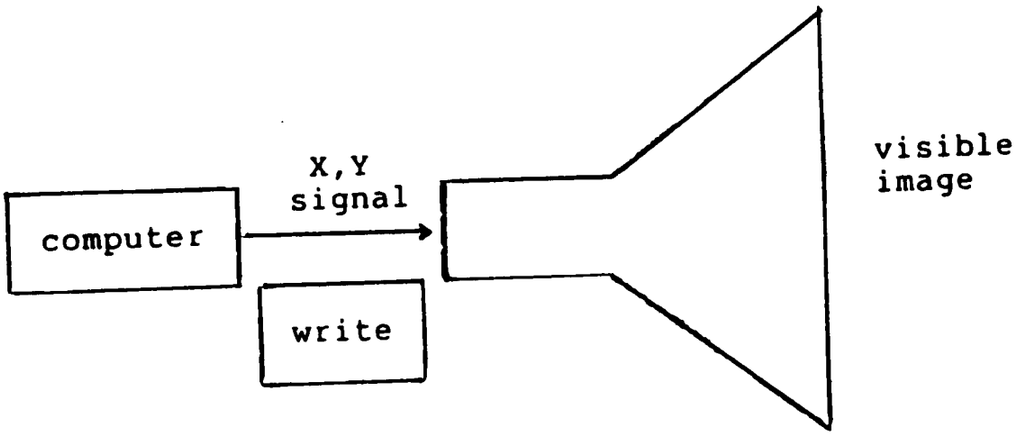


Fig 3.1 The Cathode Ray Tube

The primary electron gun produces the writing beam that strokes the graphic pattern on the phosphor screen. The secondary cathode continuously emits a diffuse stream of low energy electrons known as the flood beam, that causes the stored pattern on the screen to glow.

A single sweep of the beam produces a trace that is only momentarily visible because the phosphor lacks persistence. This means that the same pattern must be refreshed, that is, traced out repeatedly, in order to produce a steady picture. Because refreshing is time consuming, a different method known as stroke-writing refreshed display, is used with interactive graphics which require fast responses. A stroke-writing refreshed display system is depicted in fig 3.2.

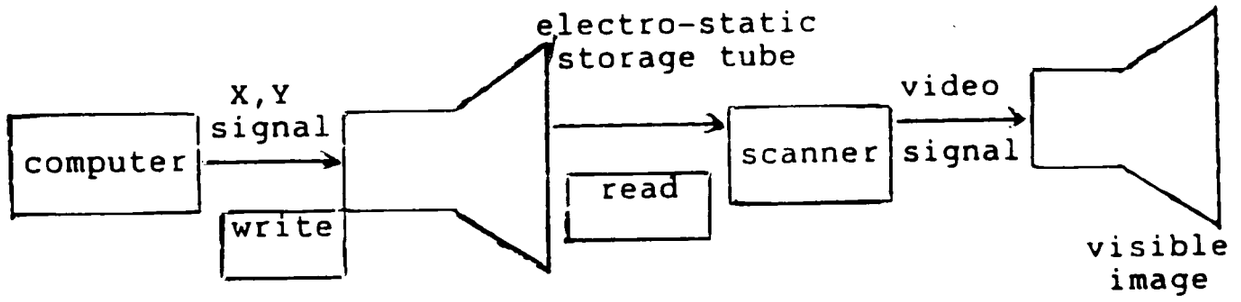


Fig 3.2 Stroke-Writing Refreshed Display

An interactive graphics system* builds a complete picture, and a display file that holds the current view of the picture. The display file is placed in refreshed memory(a storage area in either main memory or within the graphics terminal). The display processor(DP) reads the contents of refresh memory and sends instructions to the vector and circle generators, which convert the geometric descriptions into XY analogue voltages to control deflection of the electron beam. The entire image is stroked during the refresh cycle. A refreshed display does not have an erase operation; instead the interactive graphics system changes the image by replacing the contents of the display file.

3.2 Raster or Bit-Maps

The digital data stored in refreshed memory are the basis for the video signal in computer graphics. The simplest data that describes an image is one bit for each pixel in the image. The data are stored as an array known a a bit-map. One dimension of the array is the number of

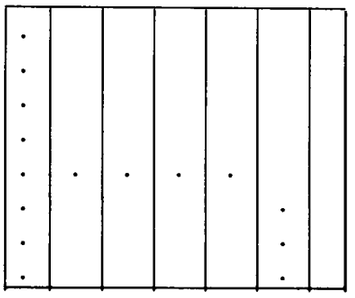
* a graphics system is defined as a computer system with some sort of display

horizontal lines, and the other is the number of pixels per line. Typical array sizes are 512 X 512, 256 X 256, 1024 X 1024. The computer can change the value of any bit independently, making each pixel an addressable point.

The resolution of the DP is the number of addressable points in the array. Obviously, the higher the resolution the better the quality of the picture.

3.3 Drawing Text & Lines by Dots(Pixels)

By displaying dots in a rectangular array pattern such as the one in fig 3.3, it is possible to produce a display of text. There is no single, widely accepted size for the dot array that defines a character, but 7 X 8 and 6 X 8 are common.



The letter h in a
7 X 8 dots array

Fig 3.3 Displaying Text by Dots

To display a whole string of text, the dots for the string of text may be held in a list.

Lines on the other hand are plotted by displaying a sequence of points. Several algorithms for line drawing exist. To visualise the concepts of line drawing in computer graphics we will consider only one of them, the differential form.

The differential form assumes that the equation of a line is represented by a differential equation as is explained below:

Suppose we want to draw a line from (X1,Y1) to (X2,Y2)

Let

delta-X = X2-X1

delta-Y = Y2-Y1

Then

dy/dx = delta-Y/delta-X

Solving this differential equation with the given

boundary conditions, gives:

X(n) = X(n-1) + 1 with X(0) = X1

Y(n) = Y(n-1) + delta-Y/delta-X with Y(0) = Y1

So X increases by one each time, and Y by delta-Y/delta-X

Since Y-display must be an integer, then

Y-display = trunc(Y(n))

That is, truncate the value of Y(n) to get Y-display

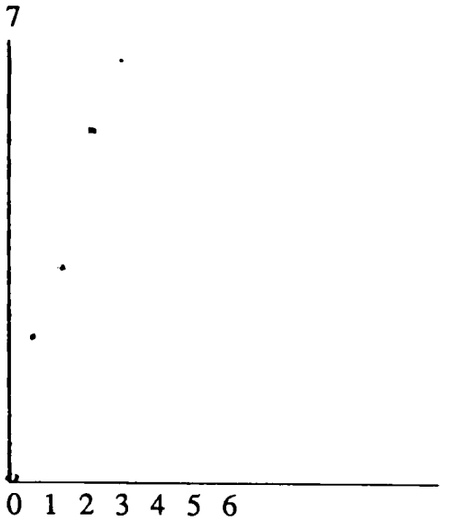
To get better values for Y-display it is better to use

Y-display = trunc(Y(n) + 0.5)

For example consider the line from (0,0) to (4,6). The (x,y) pairs obtained by this method are shown in table below:

X(n)	0	1	2	3	4
Y(n)	0	1.5	3.0	4.5	6.0
Y-display	0	2	3	5	6

The plot of these points is illustrated below.



Since these points are discrete, the computer graphics line is not as straight as the mathematical line and it has a size(size of dots).

3.4 Graphics Packages

A display file is effectively a program for execution by the DP. Although the display file can be created by writing down the instructions to represent a desired picture, they are not created this way. Instead a computer program is written to generate the display file. This is called display file compilation because a program is generated by running another program (compare with program compilation).

To illustrate display file compilation, consider the actions required to display a graphic function:

LINE(100,150,500,600)

This is a request to add the line from (100,150) to (500,600) to the display file. The resultant instructions are as follows:

```
    set X     100 < --- old pointer
    set Y     150      position
set delta-X    400
set delta-Y    450 < --- new pointer
                     position
```

The set of functions that constitute the display file compilation may be used with a variety of different application programs. They may be grouped as a package of subroutines(graphics package) that application programmers load in programs.

3.5 Data Structures for Computer Graphics

In[39], Williams surveys data structures and their use in computer graphics. After surveying several graphics systems, he points out that linked lists like ring structures, trees and network structures are widely used and can support complex graphics systems. It is further shown that more flexible structures like hierarchical ring structures are the best. Flexibility, however, also poses the biggest problems. But the irony is that the more flexible a structure is, the more difficult it is to implement. Also, the more flexible a structure is, the more pointers it uses, and therefore, the more memory space it requires.

Because of the above mentioned problems with linked structures, other general structures have been tried. Williams in[40] describes how relational data bases can be used in graphics applications, and how user functions can be constructed as operations on relations. He also showns that the use of relational data bases simplifies the handling of data but with some loss of machine efficiency. Foley and van Dam in[34] also support this view. But further adding that in using relational data bases "...there is no need to build and access linked lists or any other structures. Instead the programmer can build on top of a general purpose facility by defining tables of relations. Queries can be formulated to retrieve special n-tuples which is similar to linked list pointer chasing, or enumerate all associations in a relation which is similar to looping through an array or a linked list".

3.6 Computer Graphics Input Devices

A graphics input device captures data and makes them available to the graphics programs. Many types of input devices exits, in this section a summary of the main features of some of these is given.

(1) Alpha-numeric Keyboard

All graphics terminals incorporate a full alphanumeric keyboard as a standard feature. However, keyboard techniques are not ideal for positioning and selecting. The keyboard is generally used to enter non-graphic data in the graphics system.

(2) Light Pens

The main function a light pen is to point at graphics objects shown on a refreshed CRT. The positional input called tracking, is accomplished by moving a cursor. A simple opening in the tip of the pen permits entrance of light. The light entering reaches an electric detector that reacts by generating an electrical pulse. This electrical pulse is detected and used to determine the position of the object.

(3) Digitisers

A digitiser measures the position of a point within a two dimensional area and codes the measurement into XY coordinates, suitable for computer processing. The device consists of a flat surface, the digitiser tablet and a positioning tool like a stylus or cursor or mouse.

Many variations of digitiser exists, some of these are:

Touch-sensitive digitisers

These use an ordinary pen in conjunction with a touch sensitive surface. The surface senses pen movements as XY values

Sonic digitisers

Sound impulses generated by an electric spark at the point of the pen are detected and used to determine the location of a point.

Light Detector digitisers

These equipment resemble a picture frame. A number of tiny light-emitting diodes(LEDs) are installed along one horizontal and one vertical edge of the frame.

Light detectors are attached to the two edges opposite the LEDs. The rays of infra-red light created by the LEDs is interrupted when the operator places a finger or pen on the digitiser area. The photo-detectors signal the interruption to the digitiser's logic, which generates the coordinates of the touched point.

(4) Joysticks, Trackballs & Dials

These are used mainly for cursor-symbol control instead of a light pen. Each of these devices has two components, a handle and a sensor. (the only difference about them is the type of handle used). The sensor passes input data in suitable digital form to the graphics system.

(5) Buttons & Switches

Other input devices include buttons and switches. A switch is a input device that remains in one of the two possible states until changed. A button is a special type of momentary switch that rebounds after being pressed. A message is sent to the program when a button is pressed or a switch is reversed. This message indicates the fact that data entry has occurred.

3.7 Computer Graphics Output Devices

These are devices that allow one to retain lasting results of the picture generated by a graphics system.

(1) Pen Plotters

A digital pen plotter is a servo mechanism, an automatic feedback control system for mechanical motion. Several types of these exists.

Flatbed Plotters

With these plotters the pen moves in x and y directions on a flat surface while paper is held by an electrostatic charge.

Drum Plotters

The pen changes in x and y directions relative to position of paper, but it does so by a combined movement of the paper.

Beltbed Plotters

These are a cross between flatbed and drum plotters. A continuous belt connecting both sides of the flat vertical plotting surfaces moves in conjunction with the pen.

Plot Back Plotters

A plot back plotter is a hybrid device that combines a flatbed pen plotter and a digitiser. It can be used as a plotter or as a digitiser, to input the coordinates of the pen into a graphics program.

(2) Electrostatic Plotters

These depend on applying electrostatic charge to the paper, then either:

(i) the matrix-writing techniques or

(ii) the photoconductive plate methods are used

(i) Matrix-writing techniques

Produce an invisible image on the surface of the paper with a static electric charge and then applying a liquid toner containing suspended carbon particles that become attached to the charged areas to create a visible image.

(ii) Photoconductive plate methods

The image is produced in an internal CRT. The visible light from the CRT is transformed by a photoconductive plate into a charge on paper. After an exposure time of several seconds the toning and drying process is applied to the entire page and a visible image is obtained.

(3) Graphic Film Recorders

A graphic film recoder is simply a camera that photographs an internal CRT.

3.8 The CORE Graphics System

Recently there has emerged a proposal for a graphics standard, the Graphics Standards Planning Committee's CORE system[41]. The CORE system was developed to make graphics programs more portable. To this end it strives for device independence. It also aims to specify only the basic graphics capabilities, providing a foundation for more advanced techniques. Both these objectives make this standard a good model[42].

The CORE graphics standard is not the only graphics standard available, there are others. For example the Graphical Kernel System(GKS)[43] is widely used.

Since the full establishment of data base technology, integration of graphics based information with data bases has been vigorously sought. In relation to DBMS, graphics based information has been used in three main areas:

(i) graphics interfaces, discussed in chapter 4

(ii) storing picture-based data, discussed in chapter 5

(iii) in multimedia databases, discussed in chapter 6

Chapter 4

User Interfaces to Data Bases

4.1 Why User Interfaces?

In the ANSI/X3/SPARC report[28], an interface is defined as "a language accepted by two(or more) processes for describing data communicated between them". From this definition, a user interface is, therefore, a program that allows man and computer to communicate. But, why user interfaces? The need for user interfaces is explained better by Penney[44] in the following way: "The computer is a difficult beast to ride. The source of difficulty lies not in the operating system nor in the organisation of data; though complex, these are things that can be learned. As technology changes, they may require to be re-learned, but the problem is still one of mental capacity and assiduity. The real problem is at the point of contact between the human being, director, manager, clerk, etc and the computer system".

Ever since people started using computers, much effort has been spent in finding ideal user interfaces. Details of factors affecting man-computer communication will not be discussed in this work. For these details, see[45, 46, 37].

With the firm establishment of data base technology, and the wide-spread use of data bases in the offing, it was soon found out that most users of data bases will be not professional computer staff but the casual user, the novice. Codd in[47] and in[48] explains the issue in more detail. The casual user only uses the computer occasionally, so he need not learn the insides of a schema or data dictionary. In short, he is not interested to know the structure of the data base but only to get data from it.

The need for data bases to be used by novices, makes the problem of designing "user friendly" interfaces of paramount importance.

4.2 Command Language Driven Systems

Data bases are normally accessed through some form of a query language. The user issues request commands to the DBMS, and which in its turn does what the user requested it to do. For relational data bases where no navigation is required to get data from the data base this does not present any problems, however, for the network and hierarchical data bases this is a problem. In such cases higher level commands in the form of programs are provided by professional programmers[24].

To be user friendly, a command language interface must ideally be natural language (because this is the language humans naturally use). But, as is well known, immense problems exist in making computers understand natural language[49]. Therefore, few natural language interfaces to data bases exist, RENDEZVOUS developed by Codd[48] is a good example. As an alternative to natural language, English like formal languages have been designed in the hope that these provide simple-easy-to-learn means of expressing primitive actions to obtain information in data bases. Structured English QUEry Language, SEQUEL[50], (later renamed SQL) which was designed for people who have need for interaction with a large relational data base but who are not trained programmers, is a good example.

Using the COUNTIES relation described in figure 2.2, formulation of the query:

find the names and sizes of counties whose

populations are greater than 100,000

will appear in SEQUEL as

```
SELECT NAME SIZE
    FROM COUNTIES
    WHERE POP >  100000
```

Using some of more mathematically oriented query languages which use range variables like QUEL*, the formulation is:

* QUEL is the query language (relational calculus based) for the relational DBMS INGRES

```
Range of x is COUNTIES
Retrieve (x.NAME, x.SIZE)
where x.POP >  100000
```

The advantages of SEQUEL in terms of readability and ease of formulation compared to languages which use range or existential or universal quantifiers is obvious.

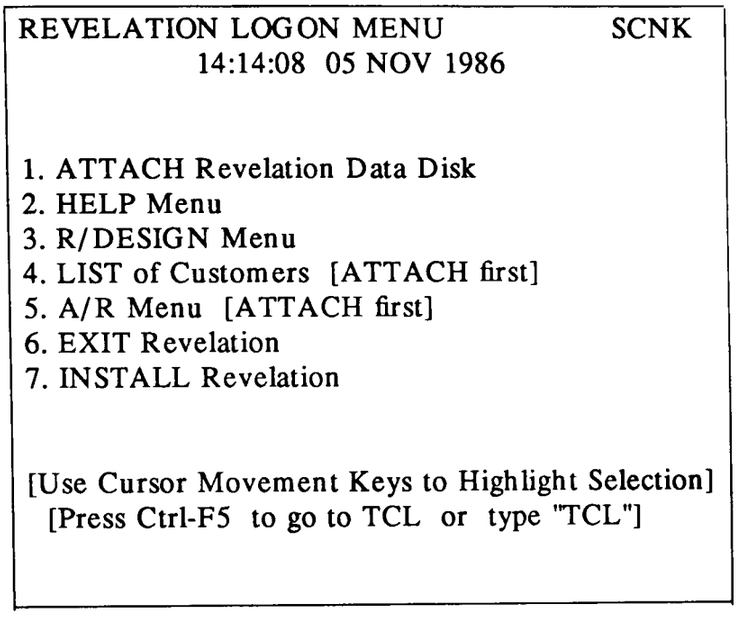
However, the main drawback of command language interfaces is that the command prompt provides little information as to the commands available and the format for using such commands. Users must remember the syntax of the command and enter it directly from the keyboard each time the function needs to performed. This intimidates novice users. What intimidates them is that they have to take an action. They have to know ahead of the time what commands they need.

Below are some of the other methods of making user friendly interfaces by providing pleasant graphic interfaces and other methods rather than the sparse command line.

4.2 Menu Driven Systems

An operation menu is a list of operations from which one selects one. A menu driven system is therefore a system where the user is given an operation menu to choose from.

When compared to command driven systems, computer menu systems are appealing because they reduce memorisation of commands, reduce training, and structure the user's decision making[51]. Figure 4.1 is the main menu a user sees when he/she logs into the Revelation data base. It is immediately clear from the menu, which commands are available to the user. Together with the accompanying explanations, the user can easily choose what he or she wants to do.



Do process number 1 before choosing option # 4 (LISTing)
or # 5 (A/R Menu)
F5= Toggle MAIN/LAST menu Ctrl-F5= TCL
F9= END menu Return= Run menu option

Fig 4.1 Revelation Main Logon Menu

Figure 4.2 is the menu displayed if the user chooses option 2 in the main login menu. Again, it is clear to the user the options open to him or her at this stage.

REVELATION C O M M A N D S 14:16:38 05 NOV 1986

1. ATTACH	18. EDIT	35. SELECT
2. BASIC	19. FORM	36. SET-COLOR
3. BLIST	20. FORM-LIST	37. SET-CRT
4. CATALOG	21. GET-LIST	38. SET-FILE
5. CHANGE-PASSWORD	22. LIST	39. SET-LPTR
6. CLEAR-FILE	23. LISTFILES	40. SET-OPTIONS
7. COMPILE	24. LISTMEDIA	41. SORT
8. COPY	25. LOGTO	42. SSELECT
9. COUNT	26. LOOKDICT	43. SUM
10. CREATE-ACCOUNT	27. MOVE	44. TERMINAL
11. CREATE-FILE	28. NAMEMEDIA	45. TEXT
12. DELETE	29. PORT	46. TIME
13. DELETE-ACCOUNT	30. RDESIGN	47. WHO
14. DELETE-FILE	31. RECREATE-FILE	
15. DELETE-LIST	32. RENAME-FILE	
16. DETACH	33. RUN	
17. DUMP	34. SAVE-LIST	

Enter COMMAND reference number,"?" or "?number"

Fig 4.2 Follow on Menu After Option 2 on Main Menu is Chosen

Figures 4.1 and 4.2 show what are known as explicit menus, menus which supply an explicit enumerated list of items from which the user selects by typing a number or letter. A variant to this theme highlights or capitalises the first letter of the selectable item. The use of icons is also a form of explicit menu.

While explicit menus posses obvious advantages over command driven systems, in some situations explicit menus can themselves be inefficient. Embedded menus[51], where the menu items are embedded within the information being displayed in some respects represent an improvement. In embedded menus, highlighted or underlined words or phrases within the text become menu items, and are selectable by touch screen, cursor, or mouse methods. Figure 4.3 is an example of an embedded menu in which where selectable items are in bold capitals.

a database management system, **DBMS** of which
INGRES is an example, is a program that
coordinates several different users' access
to a collection of data.
A DBMS is important software for a computer
system. While balancing the different
requirements of different users with respect
to the same collection of data, a DBMS is the
tool that helps users manage their data, too.
The term **APPLICATIONS PROGRAMS** refers to programs that
solve real-world problems. Application software
contrasts with **SYSTEMS SOFTWARE**, which solves
problems that concern the computer.

Fig 4.3 Example of Embedded Menu

Embedded menus come in very handy in spelling checkers, program editors, and interactive graphics systems where explicit menus potentially become very verbose, so their use will take a lot of space on the screen.

4.4 Forms Driven Systems

Forms are an electronic equivalent of paper forms. Figure 4.4 is an example of a form.

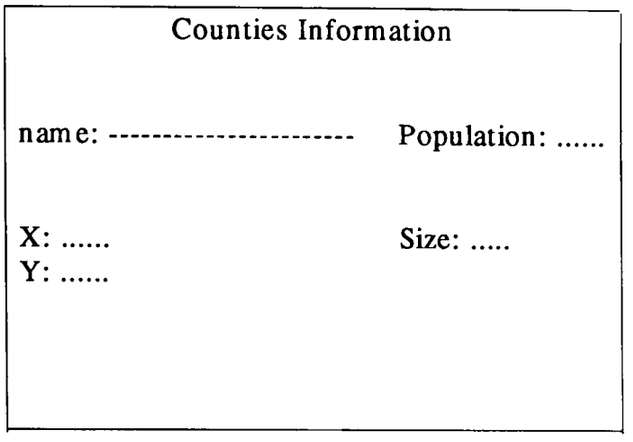


Fig 4.4 An Example of a Form

Forms assure ease of use in a variety of tasks because they enable a wide range of activities to be performed without requiring that you know a special language or learn lots of complicated procedures. In the INGRES product[52], use of forms is called Visual Programming because forms work is a highly visual process. With forms, one interacts with structures at various positions on the terminal screen, employing tools that are simple and intuitive.

Forms driven systems are driven by frames. A frame is a combination of a form and an operation menu, figure 4.5 is an example of one.

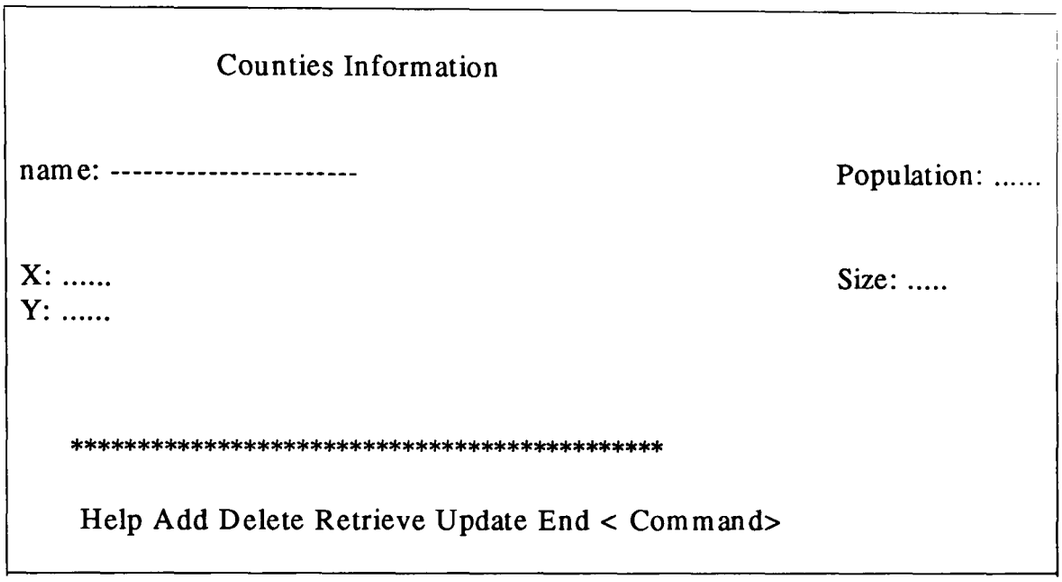


Fig 4.5 An Example of a Frame

Frames can obviously have a variety of appearances. However, regardless of the differences between frames, the techniques for making use of them is the same. These techniques fall into the following few basic categories:

(1) Moving around the frame

Most terminals offer "arrow keys" that move the cursor from field to field in the frame.

(2) Entering or editing data on the frame.

There are two basic instances in which data is entered onto a frame:

(i) entering data into a field, and

(ii) editing existing data in a field.

Whether it is entering or editing data, however, the fundamental procedures are the same.

By use of the cursor keys, the cursor is moved to the required field and overstrike typing is performed. This means that whatever text is typed it is accepted as new data.

(3) Selecting and Executing Menu Operations

After entering data values, for example, one may want to add them to the data base. By use of the cursor keys, the name of operation (eg ADD) may be selected and executed.

4.5 Graphics Based Interfaces

It is widely acknowledged that pictures are superior in communicating information than pure text. Because of this broad communication bandwidth, the potential for computer graphics to act as friendly user interfaces was spotted early in the development of computers. It was expected that, the power of computers to manipulate information, and the power of graphics to communicate information will forge an ideal combination. This natural combination is explained clearly by Ingalls in the following sentence[53]: "Qualitatively, people think with images, and any system that is incapable of manipulating images is incapable of augmenting such thought. Quantitatively, a person can virtually absorb information equivalent to millions of characters per second, while the normal rate for reading text is less than 100 characters per second". Foley and van Dam[34] support this view when they say that interactive graphics is "the most natural means of communicating with a computer".

The development of Sketchpad[36], and its success in handling computer-human communication, enhanced the use of graphics in computer-human dialogues. Several user interfaces which have exploited this feature are worth mentioning. These are Cupid[54], Zloof's Query-by-Example(QBE)[55], and Ski[56]

4.5.1 The CUPID System

In the CUPID(Casual User Pictorial Interface Design) system, the user accesses information in a data base by writing queries by aid of pictorial symbols. Instead of writing text sentences, the user draws diagrams on the screen using specially known symbols for relations, items, operators, etc.

CUPID is built on top of the relational DBMS, INGRES and the pictorial symbols are compiled to an equivalent QUEL statement.

Using the relation COUNTIES shown in figure 2.2, the example query shown below is represented in the CUPID pictorial language in figure 4.6.

Query:

write down the names and sizes of counties whose populations are greater than 100,000

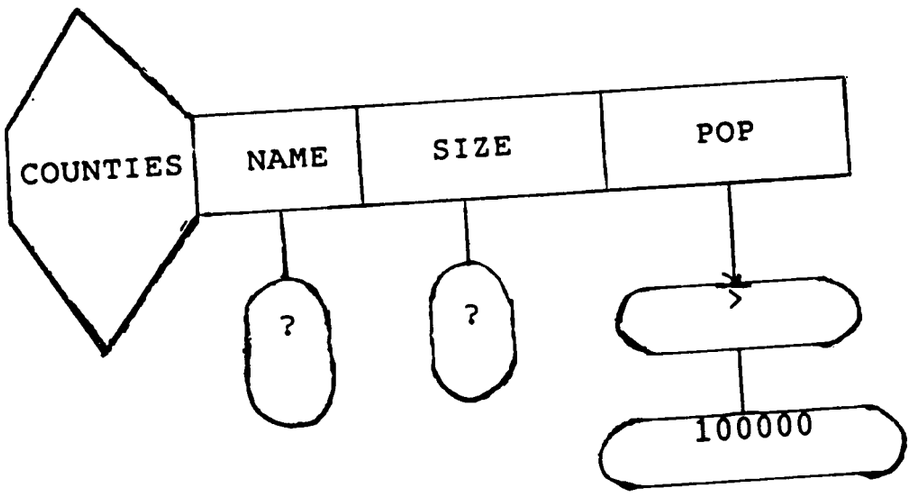


Fig 4.6 CUPID Example

The advantages of this type of syntax are:

1 no unbounded text strings

2 only constants are entered from the keyboard, thus eliminating most spelling and typographic errors

3 user friendly

This makes formulation of queries easier and clearer.

4.5.2 Query By Example

Zloof's QBE on the other hand allows the user to access a data base by filling forms displayed on the screen. The basic idea is that the user formulates the query by entering an example of a possible answer in the appropriate place in an empty table. Each operation is specified by using one or more tables; each such table is built up on the screen, with column names being supplied by the system and the other parts by the user. Example elements which are indicated by underlining are variables specified by the user and solved by the system during query processing.

Using the COUNTIES relation described in figure 2.2, figure 4.7 is formulation of the query:

Query:

write down the names and sizes of counties whose populations are greater than 100,000

COUNTIES

NAME	X	Y	POP	SIZE
P.		> 100000	P.	

Fig 4.7 Query-By-Example Example

By typing the relation name COUNTIES on the empty form given by the system, the system automatically fills in the column names. the P in the form instructs the system to print those columns and in the POP column the > 100000 value typed is not underlined, so only tuples with this value will be printed.

It is appreciated that this form of accessing the system enables the user to formulate queries without resorting to extensive use of dictionaries because the system also supplies some information to the user when formulating a query instead of leaving everything to him. Thus,

QBE enables even non-experienced users get information from the data base.

Another form-based query language similar to QBE (but used to query pictorial data bases) is Query-by-Pictorial-Example(QPE), this is discussed in section 5.4

4.5.3 The Ski System

Ski(stands for A Semantics-Knowledgeable Interface) was intended to be a naive user interface, but at same time allowing access to the full facilities provided by the DDL and the DML. During a session, the user is presented with a formatted screen. Figure 4.8 is a simplified example of such a screen with only one relation appearing. On choosing a function, the screen will change to display the requested results.

The screen is broken into a variable number of strips. The top strip represents one or more parent types, which may be any of the types in the schema. The next strip represents attributes of the parent types, etc. The screen therefore provides a medium for representing the complicated structure of the schemas.

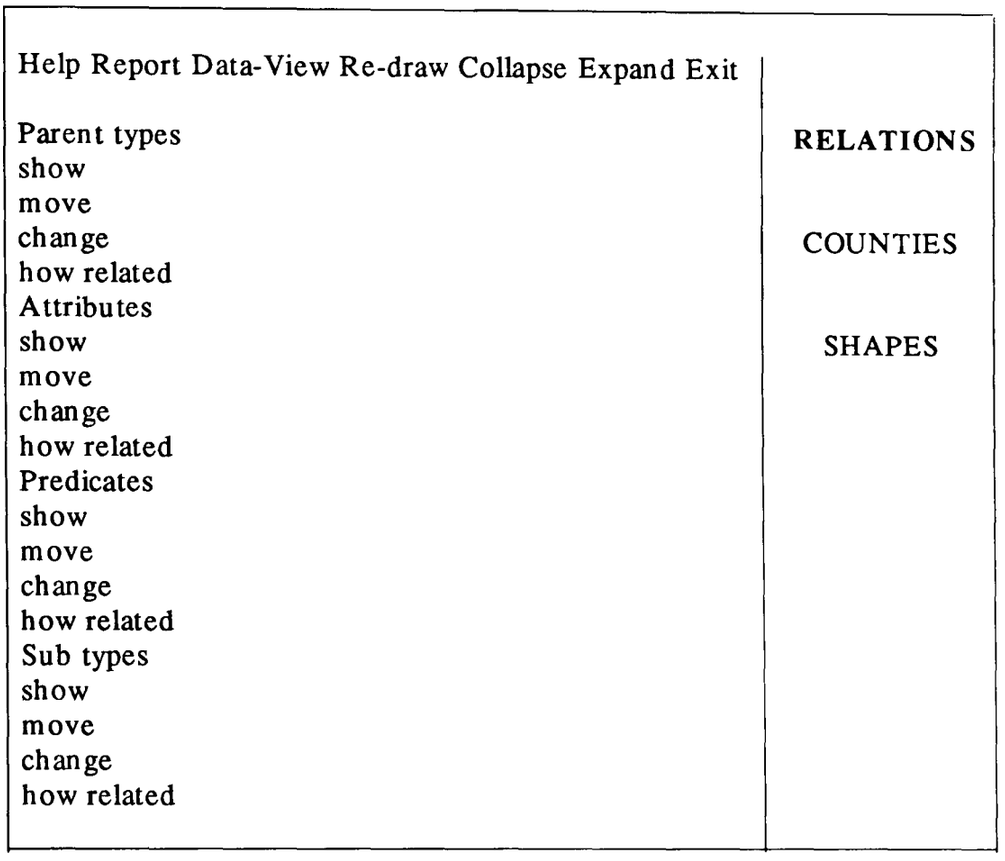


Fig 4.8 Ski Formatted Screen Example

The user uses a mouse to control and choose Ski operations. While interacting with Ski, the user uses the Ski operations to create a session view. The user selects schema components of interest and peruses the schema for related information. The unique feature of Ski is that this perusing is not performed navigationally but semantically.

4.5.4 The Spatial Data Management System

Another interesting graphics based user interface is Herot's Spatial Data Management System(SDMS)[57]. This system is really an INGRES DBMS running under UNIX† and can be accessed as a normal data base. The system, however, has the ability to compile data stored in relations to pictorial form and store the pictures on a video disk using ready made icons, see figure 4.9.

† UNIX is a trademark of Bell Laboratories.

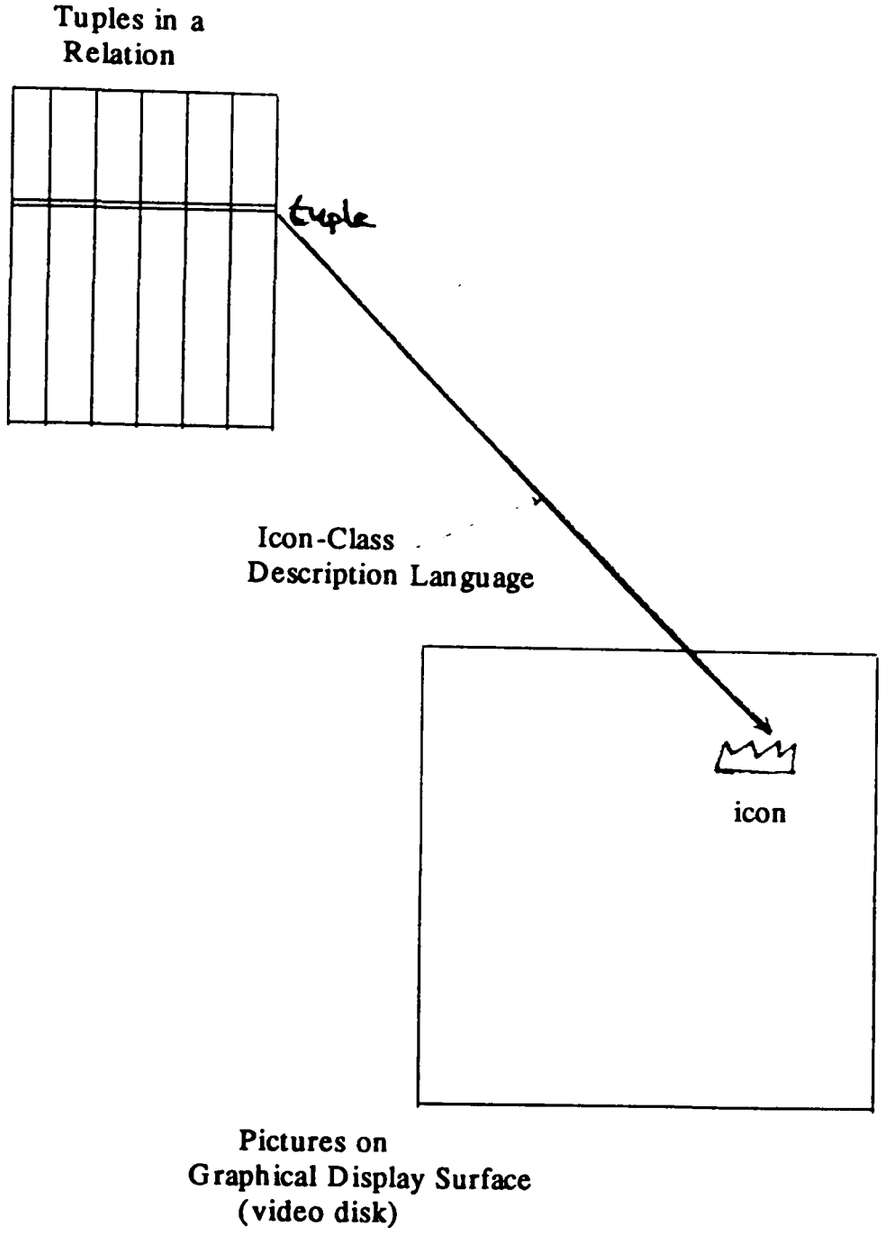


Fig 4.9 Text Data to Pictorial Data Compilation

The user can process (update, query, etc) either of the two representations of the data base; the data in tuples or the pictorial data on the video disk.

To view the data, the user uses a joystick to browse through the data on the video disk. The system also allows the user to zoom in on any data that is to be examined in more detail, making it appealing to both advanced and novice users. Zooming also encourages the user to move through the data, getting more details without resorting to the use of any dictionaries.

4.6 Summary

This chapter has presented several ways of constructing user interfaces in human-computer communication in particular how to access information in relational DBMS.

Data bases are normally accessed through some form of a query language in the form of commands. The user issues request commands to the DBMS, and which in its turn does what the user requested it to do. However, the main drawback of command language interfaces is that the command prompt provides little information as to the commands available and the format for using such commands. Users must remember the syntax of the command and enter it directly from the keyboard each time the function needs to performed. This intimidates novice users. What intimidates them is that they have to take an action. They have to know ahead of the time what commands they need. To overcome the shortcomings of command languages, other forms of interfaces have been tried. These are menu and graphics-based interfaces.

There has been wide use of menus in many aspects of human-computer communication. This is mainly because menu based systems are appealing (especially to the novice user) because they reduce memorisation of commands and their formats. The other factor making menus based systems appealing is that they can be programmed to be 'foolproof' in the sense that the programming denies the user any chances of performing unauthorised actions. For a system that is to be used by non-expert users this is also a very good feature.

The major drawback of menu based systems is that for the expert user frequent traversals of familiar paths may become cumbersome. An alternative may be the offering of shortcuts or command languages as a means of bypassing certain menus.

The motivating factor in the design of graphics based interfaces is the belief that pictures enhance communication. Of the graphics based interfaces discussed in the chapter, Query-By-Example has been successful and its ideas have been used in other systems like Query-by-Pictorial-Example and Query-by-Structure-Example. These are discussed in sections 5.4 and 5.6.3 respectively. The ideas in the CUPID system have not been well received. No follow up

work has been done nor similar system has been designed. This can possibly due to the complicated way in which the pictures in formulating a query have to be structured. CUPID also does not have a help mechanism to the user. The user has to remember relation names and domain names. This is opposed to the QBE system where when the user fills in the relation name, the system helps him/her by providing the domain names.

The use of graphics in the design of interfaces has increased the ways of constructing user interfaces. Graphics based iconic menus, embedded menus and pop-up menus are widely used now. Such interfaces are used in conjunction with a pointing device, for example a mouse. Use of such menus has two advantages. First, it cuts down dramatically the number of key-strokes the user has to enter. Second, the interaction is highly visual. This makes the human-computer communication much more appealing, therefore much more user friendly.

Chapter 5

Storing Picture-Based Information in Relational Databases

5.1 Introduction

The potential for applying computers to large masses of pictorial information, such as remotely sensed earth resource data or medical photographs, leads to the necessity for storing pictorial based information in data bases[58, 59].

The use of relational data bases to store pictorial data is much more appealing because it frees the data from any specific storage structures, thus providing means for extending the lifetime of pictorial data by divorcing it from particular hardware. Furthermore, it allows integration of shared data data bases that includes symbolic and numerical information[60].

The other motivation for storing picture based information in relational databases is the simplification obtained in handling data structures in computer graphics. See section 3.5, where this issue was discussed at some length.

Several systems have been designed to assess the feasibility of storing picture based information in relational data bases. In this chapter, a survey of such projects is made. The systems discussed primarily store, manipulate, and retrieve the picture-based data and not the pictures themselves. The pictures are generated on the display when the corresponding data is retrieved and plotted. However, Lien and Utter[61] describe how they designed an integrated Image Database Management System, the IDMS, which allows users to store, retrieve and manipulate images. The system however, uses no database concepts in the normal sense of DBMS.

5.2 The "Picture-Building" System

A project by Williams and Weller[62], reports on the ability of storing both graphical and non-graphical data in relational data bases. They report on the Picture-Building-System, a

system with a mechanism for viewing and interactively manipulating the graphical data via pictures.

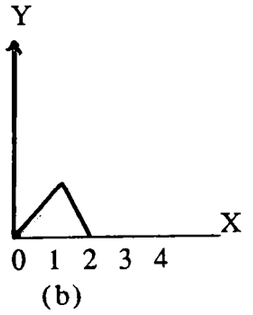
In this system, both graphical and non-graphical data are stored together. For picture building purposes, domains have graphical meanings. For example in a relation there might be a domain for COLOUR, INTENSITY, SCALE, X,Y, and Z(for position), etc. The system also has a set of allowed XYZOPERATIONs for graphics functions, for example MOVE, LINE. AREA, etc.

Figure 5.1(a) is an example of a relation that draws lines to form the triangle below it.

TRIANGLE

X	Y	XYZOPERATION
0	0	MOVE
2	0	LINE
1	2	LINE
0	0	LINE

(a)



(b)

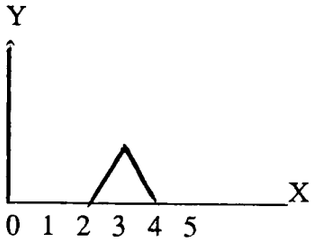
Fig 5.1 Triangle Definition

Figure 5.2(a) is another relation that calls the relation in figure 5.1(a) to draw the shape (the triangle translated, that is, shifted two units to the right) below it.

SHAPE

RELATION	SHIFTX	SHIFTY
TRIANGLE	2	0

(a)



(b)

Fig 5.2 Translated Triangle

Such drawing is done by the graphics interpreter that traces through any hierarchy of relations. Thus, the interpreter draws pictures from data stored in relations by use of the graphical meaning of domains and XYZOPERATIONs.

The picture-building system also contains a correlation handler and a relations editor. The correlation handler is provided to allow users to identify and manipulate displayed items. The relations editor on the other hand, allows the user to interactively create, edit, and display relations.

In addition to supporting relations as defined by Codd in[8], this system also supports (i) ordered tuples, and (ii) duplicate tuples. These features have been found to be vital when storing graphical data in relations[33].

5.3 A Relational Schema with Colour & Texture

In[60] a relational schema for describing pictures having colour and texture is described. The system also includes symbolic and numerical information.

The approach used in this system to describe pictures is to analyse the picture contents in a top down way. First the individual pictures are considered, and then the details contained in them are broken down into individual picture points. For example, figure 5.3 is a relation of a

set of snapshots/pictures. Each row, therefore, represents one picture.

SNAPS

SNAP#	DATE	PLACE	SUBJECT	NEGATIVE#	FRAME#
1	74103	MENLO PARK	CHILD	6	2
2	84115	MENLO PARK	CLIFF	8	10
3	76129	MOON BAY	SEAGULL	6	3

Fig 5.3: Example of Snaps

After describing the pictures, objects are described next. Since pictures are 2-dimensional projections of 3-dimensional objects, superposition orders need to be specified. A superposition order is some ranking of the centre of mass with respect to its distance from the observer. Figure 5.4 describes some objects of pictures in figure 5.3.

OBJECTS

SNAP#	OBJECT#	OBJECT NAME	SUPERPOSITION ORDER
1	1	CHILD	1
1	2	TREE TRUNK	1
1	3	SHADOW	2
1	4	LADY	0

Fig 5.4: Objects

In the same way that pictures are superposed of objects, objects are superposed of parts. This means that each object's details can be described considering it as a picture, so finding superposition orders on it. More precise descriptions about any part of a picture can be got by applying this process to that part. Figure 5.5 is a relation showing part superposition orders on objects from figure 5.4.

PARTS

SNAP#	OBJECT#	PART#	PART NAME	PART SUPERPOSITION ORDER
1	1	1	HEAD	0
1	1	2	HAIR	2
1	1	3	COAT	1
1	4	1	FALLEN LEAVES	1
1	4	2	LAWN	0

Fig 5.5: Parts

After describing parts, regions are specified. A region is defined as a connected set of picture points with the same colour. Colour attributes are names or values of gray level, hue, or saturation. The outline of such a region is called its boundary. Regions are the lowest level of picture description. Figures 5.6 and 5.7 illustrate regions and boundaries respectively.

REGIONS

SNAP#	OBJECT#	PART#	COLOUR	BOUNDARY#
1	1	1	PERSIMMON	1
1	1	2	DARK BROWN	2
1	1	3	SAXE BROWN	3

Fig 5.6: Regions

BOUNDARIES

BOUNDARY#	POINT#	X	Y
1	1	6.45	6.45
1	2	7.05	6.23
.	.	.	.
.	.	.	.
.	.	.	.
1	14	5.70	6.20
2	1	6.45	6.45
2	2	7.00	7.00
.	.	.	.
.	.	.	.
.	.	.	.

Fig 5.7: Boundaries

The last things to be specified in describing pictures are colour and texture. The most common standard representation of colour is in terms of tri-stimulus values Red, Green, and Blue. Figure 5.8 is a representation of colour in RGB system.

COLOUR

COLOUR CODE	COLOUR	RGB VALUE
001	SAXE BROWN	000808
002	BLUE	000006
.	.	.
.	.	.
.	.	.

Fig 5.8: Colour Codes

Figure 5.9 is an example of a relation of how texture is described in this schema.

TEXTURE

SNAP#	OBJECT#	PART#	TEXTURE#	TEXTURE NAME
1	4	1	8	TEXTURE OF LEAVES
1	4	2	10	TEXTURE OF LAWN

Fig 5.9: Texture Description

5.4 Query-By-Pictorial-Example

Chang and Fu[59] describe the form based relational query language, Query-By-Pictorial-Example, QPE, for manipulating queries regarding pictorial relations as well as conventional relations. Though closely modeled on Zloof's QBE, QPE has significant extensions to cater for pictorial operations and pictorial queries.

The QPE facilities are described through illustrative examples by considering the following relations:

ROADS(FRAME, ROID, X1, Y1, X2, Y2)

RONAME(FRAME, ROID, NAME)

POS(FRAME, XSIZE, YSIZE, XCEN, YCEN, LOC)

CITIES(FRAME, CIID, X1, Y1, X2, Y2)

CINAME(FRAME, CIID, NAME)

Query 1

Print the names of roads in the same frame as city Lafayette.

The user can express the query by making entries in four positions in two tables, as illustrated in figure 5.10.

CINAME	FRAME	CIID	NAME
	10		LAFAYETTE

RONAME	FRAME	ROID	NAME
	10		P.MAIN

Fig 5.10: Query 1 Formulation

The formulation of this question is exactly the same as its formulation in QBE. However, query two below requires the additional facilities found in QPE and not in QBE.

Query 2

Find the portion of Interstate Highway 65 that is enclosed within the city boundaries of Lafayette.

Pictorial operators are needed to finalise formulation of this query. Such pictorial operators include INT, POINT, LINE and UNION. Letters are attached to the end of the name of a pictorial operator to denote the type of the operands. The letters are P for point, L for line, and R for region. For example:
INT-LR refers to the intersection of a line and a region

Figure 5.11 is the formulation of query 2 in QPE. Three intermediate relations R1, L1 and L2 are produced in the process of solving the query. L2, the third intermediate relation is the solution.

Another category of pictorial operators found in QPE and not in QBE are those for finding perimeters, lengths of lines or curves, areas of regions, and distances between two points. Query 3 below characterises queries needing such operators.

Query 3

Find the length of Interstate Highway 65 that is enclosed within the city boundaries of Lafayette.

The solution to this query is obtained by applying the pictorial operator LENGTH-L to the result of query 2, namely LENGTH-(L2).

CINAME	FRAME	CIID	NAME
	3	5	LAFAYETTE

CITIES	FRAME	CIID	X1	Y1	X2	Y2
	3	5	X	Y	Z	W

R1	X1	Y1	X2	Y2
	X	Y	Z	W

RONAME	FRAME	ROID	NAME
	10	7	65

ROADS	FRAME	CIID	X1	Y1	X2	Y2
	10	7	S	T	U	V

L1	X1	Y1	X2	Y2
	S	T	U	V

The final solution is L2 = INT-LR(L1,R1)

Fig 5.11: Query 2 Formulation

Other facilities offered by QPE are:

(i) Image-sketch-relation conversion, whereby the user can sketch on the display data from a relation or change a sketch on the display into a relation.

(ii) Pictorial example, whereby certain desired features such as shape can be plotted and used to formulate queries in QPE, or interesting portions of displayed pictures can be pointed on the display by moving a light-pen, tracker ball, or joystick.

For example consider query 4, whose formulation is given in figure 5.12.

Query 4

Find the name of the road I pointed on the screen

ROADS	FRAME	CIID	X1	Y1	X2	Y2
	*	7	@	@	@	@

RONAME	FRAME	ROID	NAME
	*	7	P.MAIN

Fig 5.12 Query 4 Formulation

The special character "@" notifies the system that a pictorial example is being used and the "*" notifies the system that the frame number of displayed image is used.

5.5 The Image Analysis & Image Database Management System

Chang[63] describes the Image Analysis and Image Database Management System, IMAID. This is an integrated system that can store and retrieve images or image based information stored in a relational data base.

When an image is obtained, for example a LANDSAT image, it is processed by image processing and pattern recognition functions to extract descriptions and registrations. These are

stored into a relational database, while the original picture is stored in a separate image store.

To process a user's query, either data in the database tables is used or the stored image themselves.

If the extracted image descriptions are sufficient to answer the query then there is no need to retrieve the original image. This has the advantage of reducing the volume of data processed. On the other hand, if the stored information is not sufficient, all images satisfying the selection criteria are retrieved from the image storage and processed to required precision.

Data in the IMAID system is manipulated by the form based query language, Query-By-Pictorial-Example, see section 5.4.

5.6 Other Systems

5.6.1 GEO-QUEL

Go et al in[64] report the implementation of GEO-QUEL, a database with display capabilities constructed on top of the INGRES DBMS.

In GEO-QUEL, all geographic data are treated as relations, thus very little special code is needed to process the data as most tasks can be turned into interaction in QUEL, the DSL used in INGRES.

5.6.2 The Geographic Information Systems Project

The Geographic Data Systems Project carried out under a joint study between IBM Research and the County of Santa Clara designed, placed into experimental use, and evaluated a prototype computer based analysis and display system for solving problems requiring geographically related data[65].

5.6.3 Query-By-Structure-Example

In[58], a DBMS for storing and retrieving relational structures is introduced. The system is able to retrieve relational structures which are similar to an example structure or template. This is called Query-By-Structure-Example, QSE. Although QSE follows in the same analogy as

QPE and QBE, it is accessed by applications programs only, so does not prompt a user with a template.

5.7 Conclusion

The example systems presented in this chapter illustrate two very important points. First, that the relational approach to DBMS can be successfully employed to hand pictorial data. Second, that query languages need to contain special features that are capable of specifying, retrieving and manipulating pictorial data.

However, these relational DBMS systems used for storing pictorial-data are not true multimedia databases because they store attribute data only. Although these systems provide functions to draw pictures and to manipulate them on the display, the pictures themselves are not stored in the database.

Chapter 6

Multimedia Databases

6.1 Introduction

Multimedia databases, are databases which process various kinds of data:- numeric, text, image, voice, etc. The motivation to design and build multimedia databases arises because data is made of many different types. This is especially so in the office environment where documents, reports, forms, and letters come in different form[66], and more often in spoken form[67].

Most of the multimedia databases discussed do not follow the architecture of a traditional DBMS, for example, they store documents in ordinary files.
It is a pity that most multimedia databases are not implemented in DBMS in order to take advantage of services offered by DBMS such as concurrency control, crash recovery, and access control. Several reasons are given to justify this[68, 69]:

(i) Queries in this environment may be different from queries in traditional DBMS.

(ii) Queries on the image and text part of messages are not often handled by traditional DBMS.

(iii) Users of office information systems may have very diverse backgrounds from the sophisticated type to the naive. This reason is very difficult to uphold because even in traditional DBMS such diversity of user expertise exists.

(iv) DBMS have traditionally emphasised data organisation and manipulation. Presentation of data in devices with diverse capabilities has traditionally not been emphasised. This is particularly important in the presence of bit map display capabilities with different gray levels, colours, display sizes, etc.

For a longer discussion on why multimedia databases should not be implemented in

traditional DBMS see[70].

6.2 Document Processing in a Relational DBMS

In[71], Stonebraker et al propose extensions or enhancements to a relational database manager to support document processing. Proposed enhancements suggested are support for variable-length strings, support for ordered relations, support for substring operations, and support for new operators that concatenate and break apart string fields.

The aim of their proposals is not that text editing of database documents be performed using the command languages of a relational DBMS. Rather, they propose DBMS facilities that can effectively support a text editor run as an application program making calls on extended DBMS facilities.

The context in which they present their constructs is the INGRES database system.

6.3 The Multimedia Office Filing System

In[68] a system for multimedia messages is outlined. The system uses signature techniques for fast filtering, miniatures, voice excerpts and game environment for effective browsing and selection of the desired messages.

In this system, a message is defined as having a header and a contents part. When a message is received, it is labelled and stored in a separate file.

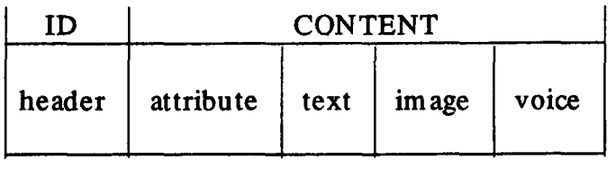


Fig 6.1 Structure of a Message

Messages are retrieved through a search of the file directories. For flexible retrieval, a combination of (i) sequential scan, (ii) keyword searches and (iii) pattern recognition techniques (for image and voice parts of the message) are used. Message retrieval facilities

consist of two parts; a filtering capability and a browsing capability. Filtering enables the user to specify what he would like to see, while the browsing capability enables the user to pinpoint from the filtered messages the ones he actually wants. The filter used is a conjunction of all restrictions on the data, text, voice, and image values of the message.

When a user is using the system, the screen layout is divided into four parts as illustrated in figure 6.2.

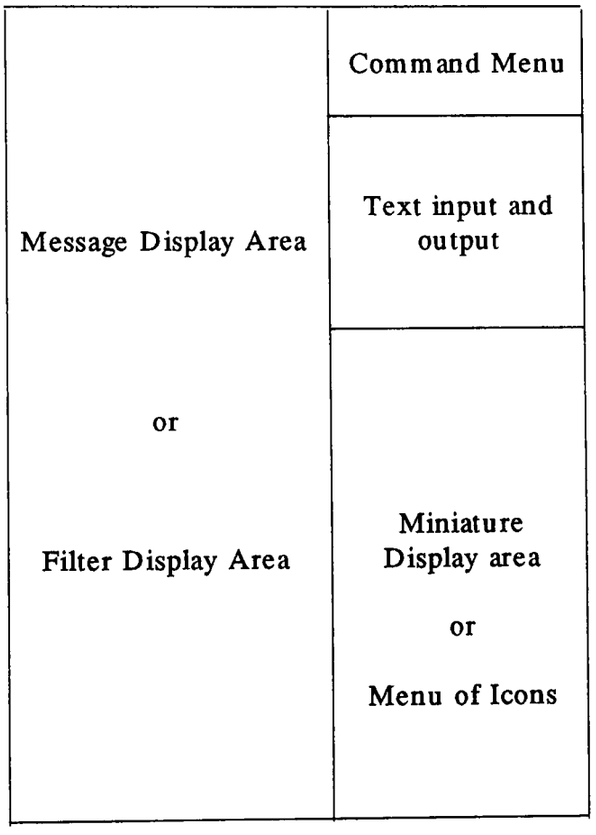


Fig 6.2 Screen Layout

The status of the display changes according to the mode. There are three modes:

(i) *create/append* - the user creates or appends to the filter

(ii) *browsing* - the user is playing the message abstractions(miniatures) for the messages that are filtered by the system.

(iii) *viewing* - the user has just frozen the browsing of the miniatures to view one of the

messages in more detail.

The Office Filing System has been implemented using UNIX on a SUN computer. The SUN computer has high resolution graphics (1024 X 1024), and a mouse facilitates input of graphical information.

6.4 A Multimedia Information System for an Office Environment

The paper[69], describes an experimental multimedia information system for an office environment being developed in the University of Toronto. In the paper, issues related to internal representation, presentation and communication with the outside world, content addressability in the various data types, user interface and access methods are discussed.

Multimedia documents in this system are known as messages, and their structure is similar to that shown in figure 6.1.

The system is accessed by the user through a screen layout as illustrated in figure 6.2. The filter used to retrieve messages can be a combination of:

a. message type

b. conjunction of attribute values and attribute ranges

c. conjunctions of disjunction of words appearing in the text part of the message

d. existence of voice

e. location of message

f. conjunctions of words appearing within the text related to the image

g. for statistical images(pie-charts, graphs, tables) the existence of attributes, attribute values

To retrieve messages, the system uses content addressability, which means that, the user specifies the content of messages that he wants to see (or not to see). This information is specified in his query. For example, the query:

Give me any documents that have some statistics on IBM.

Such a query will appear as:

.image type statistical

.IBM in text part

6.5 The MINOS System

MINOS, described in[72] is an object-oriented multimedia information system that provides facilities for creating and managing complex multimedia objects.

MINOS is very similar to the systems described in sections 6.2 and 6.3. Multimedia documents in MINOS are organised as is illustrated in figure 6.1, and the user works in an environment similar to one shown in figure 6.2. However, MINOS contains extensions for zooming, narration, transparencies, annotations, and versions.

Zooming

When the mouse is at a given point in that page, the effect of this is to expand the page at that point. For pages with images with more detail, this results in making the image more visible.

Narration

Is a voice session that is associated with a particular page of a multimedia document. It is automatically played when the user requests to see a particular page.

Transparencies

This superimposes pages. It can be very useful in overlaying two picture pages, or a picture with explanations either in sound or just text.

Process simulation

This specifies that a set of pages be turned automatically when the user looks at the first page.

Such processes may be used to explain complicated events and their interrelationships.

Annotations

These are further explanations, notes, more detailed descriptions, etc on particular documents. When annotation is selected, users can browse through the annotation in the same way in which they browse through the pages of the previous document.

Versions

This is a way of including alternative meanings to a word or text. These provide yet another way of finding information, thus increasing the man-machine communication.

The prototype of this system runs on a SUN-3 workstation running UNIX. An Instavox, directly addressable, analogue device is used to store voice segments.

6.6 The Muse System

Gibbs et al in[66] describe an experimental system that features flexible document retrieval, a distributed architecture, and the capacity to store very large documents.

This system differs from the systems discussed above because it addresses the problem of distribution.

Muse has three major components:-

a. document servers

These support the basic filing and query processing functions.

b. set of clients

These translate user operations to document server requests. These are the user workstations.

c. one server

This maintains information about network addresses of document servers and the distribution of document types.

Figure 6.3 roughly shows the display when a client is initiated. The screen is divided into six subwindows, named a, b, c, d, e, and f.

Window (a) is used to enter queries and to display document attributes.

Window (b) displays a small set of commands that are always available.

window (c) is for command selection and parameter specification.

window (d) is the display window

window (e) is for scrolling, used for browsing through a document.

window (f) is for printing system messages, error messages, help messages, etc.

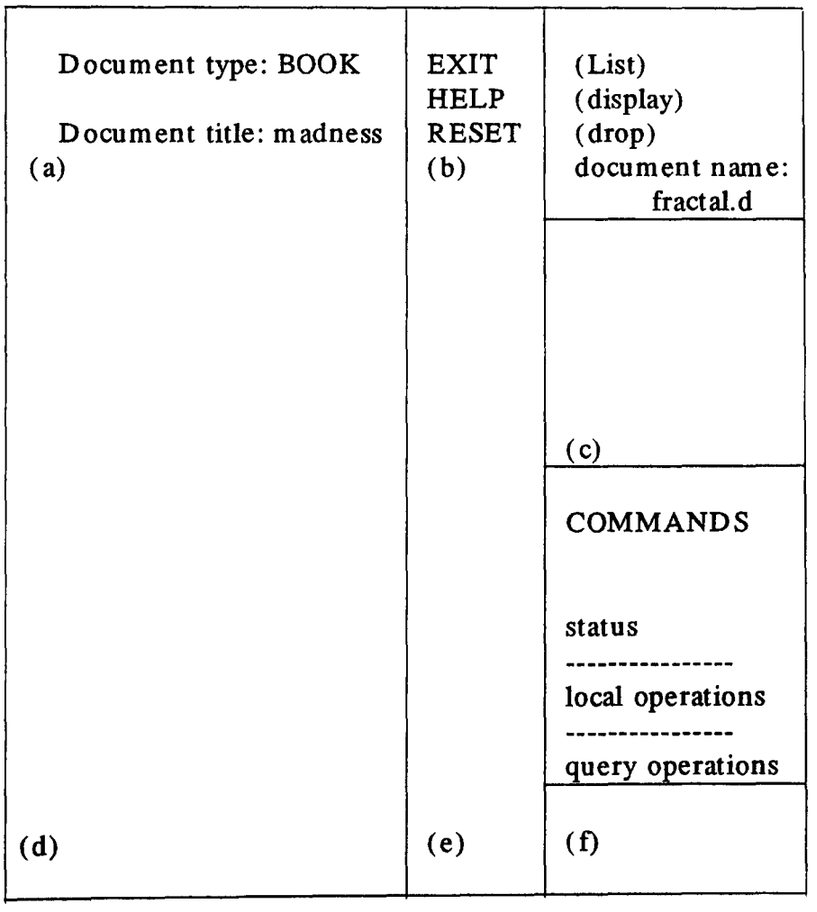


Fig 6.3 Initial Muse Screen Layout

Muse is implemented on six machines connected by ethernet; a DEC VAX 11/780, a Massomp MC-500 workstation and four SUN-2 workstations. Each machine runs UNIX.

6.7 Conclusion

This chapter has discussed four examples of multimedia database systems. These are experimental systems designed mainly for the office environment. They serve to illustrate two points. First, that the user-interfaces to these systems are far more complex than those in traditional data base systems. Second, the nature of the different types of data involved (sound, text, pictures, etc.) requires considerable thought to be given to the design of appropriate data base support facilities.

These multimedia systems, however, have one drawback. They are not true DBMS. This is because they do not follow the agreed principles of DBMS design and implementation (as discussed in chapter 2) - for example, they store data in conventional files.

Although it is not wrong to implement a stand alone data base, such an implementation misses the services offered by DBMS. Such services include concurrency control, access control, redundancy control, and indexing facilities. A stand alone data base would, therefore, need these services to be provided for in its implementation. Since such facilities can be obtained from DBMS, providing them means wastage of resources which could be used otherwise. The consequence of this is that such systems take much longer to perfect.

For example, in the MINOS so far there is no concurrency control. Thus the system is used by only one user at a given time from a SUN workstation. To control duplication a facility called document formatter is used.

Indexing in MINOS is rather subtle. The multimedia document file is a set of files organised within a Unix directory that has the name of the document. This set of files contains a synthesis file, the document descriptor, a composition file, a data-directory file, and a set of data files. The data-directory file contains the information about the various data files.

Document creation starts by creating the synthesis file. It contains the names of various data files. The composition file is also created by concatenating information in the synthesis file with the data of those data files that have been referred in the synthesis file. The document descriptor file indicates the location of the physical location where the data in the composition

file are. The document descriptor file is then the index in the MINOS.

Unlike MINOS, The Muse system can be used by several people concurrently. This is possible because Muse is a distributed system. Documents are stored in a document server and users access the information from other workstations in the local network. However, only queries can be processed on-line, updates must be done off-line.

The Muse system uses a B-tree package to build indexes with each index associating search keys(the document attributes) with physical document identifiers.

Chapter 7

Generation & Manipulation of the Pictures

This chapter discusses two issues in relation to storing bit-mapped pictures in a database:
(i) the ways and methods of generating the bit-mapped pictures used
(ii) how the pictures are manipulated or processed before and after storing them in the database. In particular, it is described how the pictures are stored in the data base files, and how they are retrieved from the data base. By treating the pictures like ordinary text fields, it is shown that it is possible to manipulate them by the normal functions provided by the data base management system.

The chapter ends with a look on the performance aspects of the picture manipulation functions. Specifically, the speed of picture processing versus efficient use of storage space is considered.

7.1 Picture Processing for Printing

Figure 7.1 is a summary of the processing used to print bit-mapped or raster pictures onto a laser printer. After generating the picture, it is stored in a file. From this file a program known as a formatter, is used to send the picture to a laser printer for printing.

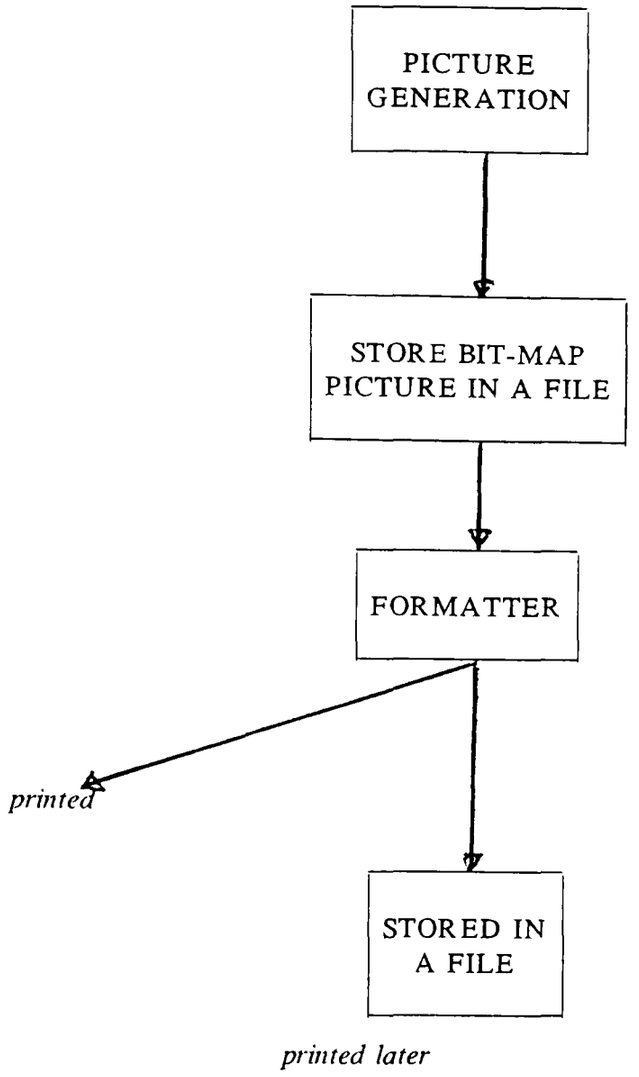


Fig 7.1 Formatting Bit-Mapped Pictures for Printing

Because the pictures are already in bit or dot form, using the 'sixel' printing protocol (see Appendix 5), only a few parameters need to be known by the formatter to dump picture dots onto the laser printers used. Such parameters include:

(i) how to initialise a page

(ii) positioning of picture on the page

(iii) size of printed picture

All these values can be obtained from the programmer's manual or technical manual of the laser printer used.

7.2 Data Used and Its Sources

GIMMS

GIMMS (Geographic Information Systems) see[73], is a user-orientated, general purpose (geographic, cartographic, graphic) processing system for use in a wide variety of applications. It is primarily of use in the analysis of geographic data by generating maps, graphs and tabular information of a thematic kind. However, it may also be used simply as a graphics package for drawing graphs of various kinds (for example line graphs, histograms, scatter diagrams, or pie charts).

7.2.1 DIGITISED MAPS

In order to assist with the mapping of geographically based data, particularly census data, NUMAC* acquired a set of digitised outline maps, suitable for input into GIMMS. Each county in England and Wales is available as a separate file. The initial maps have wards as their smallest unit, with the order of the wards within these files being identical to the order in which the wards occur in the 1981 census files currently available at NUMAC.

The files as they stand are GIMMS internal files and can only be used with GIMMS for mapping. The wards exist as AREA files with a resolution of 10 metres. Though GIMMS files are normally used internally within the package, the package provides a facility to dump all, or part of, a GIMMS file in a character form suitable for data transfer or for diagnostic analysis. Two files (GB.COUNTIES and GB.OUTLINE) were dumped and their data transferred for use in our research.

The GIMMS file GB.COUNTIES was used to generate the coordinate vectors (X,Y) for the UK counties. These vectors were then separated such that vectors for each county were put into a file.

---

* NUMAC stands for Northumbria Universities Multiple Access Computers, is a network of computers for the Universities of Newcastle and Durham plus some Polytechnics in the N.E. of England

Coordinate vectors from the file GB.OUTLINE were used to get the digitised xy points for the UK map. These were also put in a separate file.

It is data in these vector files that were loaded into Revelation files (relations SHAPES and COUNTIES illustrated in figures 8.5 and 8.6 respectively) and used to generate county maps and the UK map in this research.

7.2.2 Census Data

SASPAC(Small Area Statistical Package) is a computer package which is designed for analysis of 1981 Census Special Work-place Statistics, see[74,75]. Data files in this package are mostly viewed as one enormous table with each row in the table used for a census area and each column in the table used for a census variable. Through SASPAC, the user selects areas of interest (rows of the table) and the counts of interest (the columns of the table). The selected areas and counts may be manipulated in various ways.

Only a very small subset of this data has been used in this research, only five of the first tables. The data was summarised into counties. (that is data of boroughs in one county were added up to generate a county total for each variable). This data was put in the Smart* worksheet CENSUS and by use of the Smart Spreadsheet[76] Graphics function, used to generate diagrams, charts, and other business diagrams. See section 7.4.2.

SASPAC data was also loaded into the Revelation file COUNTIES, figure 8.6. Data in this relation can be retrieved following a user query in R/LIST. put into a worksheet and used to generate charts, and other pictures in Smart.

7.3 The Display Processor Used

Before a discussion of how the pictures are generated is made, it is better if the display processor used to display the pictures on the terminal is briefly described now. This is important because the display processor has a very strong bearing on:

---

* Smart is a trade mark of Innovative Soft. Inc.

(i) size of the pictures

(ii) quality of the pictures

The hardware supporting graphics in our case study system is the the Hercules Graphics Card. This card offers 720 pixels by 348 on the IBM monochrome display. It uses 64K of the video buffer. This 64K is divided into two approximately 32K buffers for each of the two graphics pages the card offers. The screen page is displayed on the monitor, and the RAM page is in memory. Drawing can be done on either screen, but only the monitor screen is viewable, the RAM screen is invisible.

7.4 Picture Generation

Three major systems are used to generate the bit-mapped pictures. These are:

(i) Turbo Pascal* Programs

(ii) The Smart Spreadsheet

(iii) The Microsoft Windows** Paint program.

Details of the main features of these systems are described in the following three subsections and in Appendices 2-4.

7.4.1 From Pascal Programs

Turbo Pascal programs[77,78], were written to generate the pictures. Graphics facilities were obtained from the Turbo Pascal graphics package, the Turbo Graphix Toolbox[79]. This package contains most of the functions recommended by the Graphics CORE System[41,42]. Pictures were generated either from mathematical procedures, like curve fitting, histograms, etc or from coordinate pairs obtained from digitised maps.

All drawings can be displayed either on the full screen, or in windows. Drawings can also be on a RAM(virtual) screen in memory, without display, then move the resulting images to the displayed screen when desired.

---

* Turbo Pascal is trade mark of Borland Inter. Inc.

** Microsoft Windows is a trade mark of Microsoft Corp.

Appendix 4 gives details on how to fit screens to windows, and how coordinates are mapped onto screens and windows.

The generated image is saved in a DOS file by saving into it the appropriate window contents. From this file, the picture is processed as described in figure 7.2.

7.4.2 With the Smart Spreadsheet

The Smart spreadsheet[76] is a powerful electronic worksheet. It can function alone or as part of the Integrated Smart Software System[80].

The general features of the Smart Spreadsheet are

1) a worksheet can be as large as 9999 rows and 999 columns
2) each cell can be 15 digits long, or 99 characters for text cells
3) the programs can be driven interactively by commands or through the so called "projects".

The Smart Spreadsheet provides a built-in business/scientific calculator which performs all functions available with the worksheet. These functions include mathematical, trigonometric, statistical, business, date and time, etc.

Of most interest to our work, the Smart Spreadsheet has a Graphics function capable of making:

- 2 and 3-D bar charts
- line and scatter graphs
- combination of bar and line graphs
- pie and cake charts
- area or layer graphs like
  - high-low charts
  - histograms
  - making "slide shows"

It is this graphics facility that will be used in this work to generate bit-mapped pictures

from worksheets.

7.4.3 By Microsoft Windows PAINT program

By use of a mouse, any picture or diagram can be drawn on the screen in free-hand or by use of drawing tools, with the aid of this program, see[81]. The program is able to save the picture into a DOS file. On leaving Windows, the picture in the DOS file is later processed as described in figure 7.2.

7.4.4 Comments on the Generation Methods

Each of the three methods has its own strengths and weaknesses. For example the Smart Spreadsheet method is very good for generating business or statistical charts/graphs like pie-charts, histograms, etc. On the hand, though such charts can be generated from programs, however, the process is cumbersome. Generating maps from digitised coordinates, however, is ideal for programs. Whereas, because of the free-hand nature of drawing pictures in Windows, the program is unsuitable for generating charts or maps. However, the program is ideal for editing pictures drawn in the other systems. Though the Smart Spreadsheet contains a graphics editing facility, the facility is not as good and difficult to use.

In this work the three systems are used in the following ways: although such use is not exclusive.

Turbo Pascal programs

To generate maps from digitised coordinates

Smart Spreadsheet

To generate statistical or business charts from worksheets, for example census data.

Windows Paint

To edit pictures generated from the other sources. Because it allows free hand sketching, the user can use it to put personalised headings, captions, etc.

7.5 Storing the Pictures in a Database

Figure 7.2 is the general processing done before a picture is stored into the database.

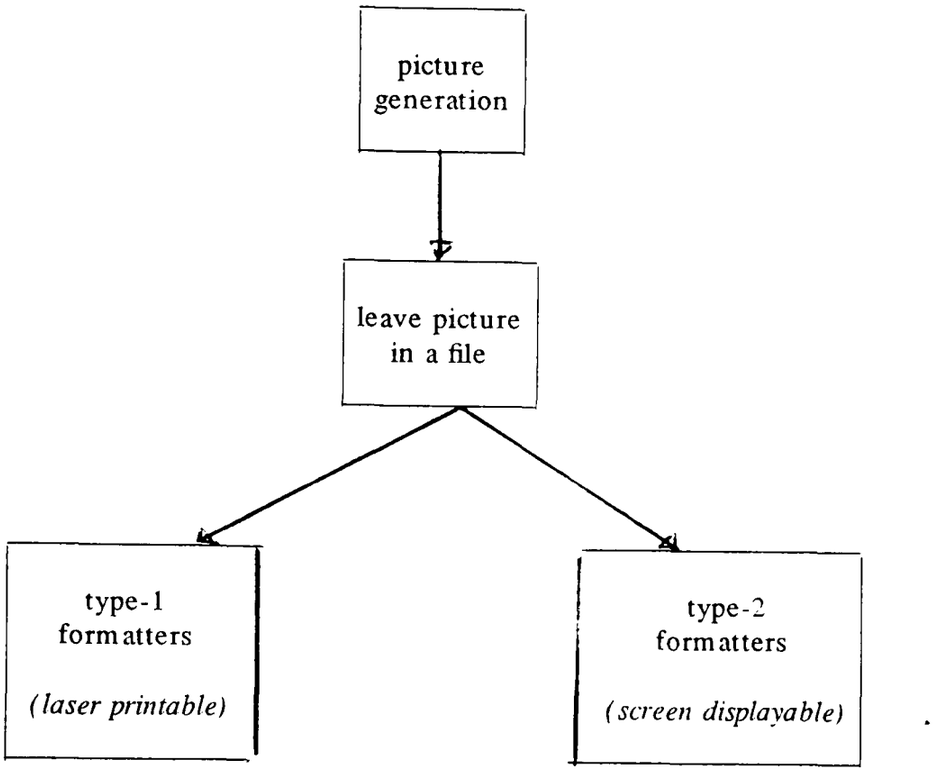


Fig 7.2 Picture Formatting for Database Storage

From figure 7.2

(i) type-1 formatters

are programs that convert the bit-mapped picture into a laser printable picture. This is the process described in figure 7.1. By supplying an aspect ratio and a scaling factor, the picture can also be arbitrarily scaled by these programs. Pictures from the different sources use slightly different formatters, but the basic functions are the same.

(ii) type-2 formatters

are programs that put a header on the picture file so that it can be displayed on the screen from a Turbo Pascal program without going back into the generating program.

In Revelation, the pictures are stored as TEXT fields. (Note that Revelation distinguishes between text fields and alphanumeric fields*) Two principal relations, similar in structure, are used. These are LASERPICS for laser printable pictures, and SCREENPICS for screen displayable pictures. The schema of these relations is illustrated in figure 7.3.

PICNAME	PICTURE

PICNAME in both relations represents the picture name and the key

Fig 7.3 Picture Storing Relations

7.6 Storing Binary Programs in Database

In addition to storing bit-mapped pictures in the database, binary programs are also stored. These binary programs are compiled from source languages such as Turbo Pascal, 'C' or assembly. Binary programs are stored in the relation PROGRAMS, whose schema is illustrated in figure 7.4. The program is stored as a TEXT field and the program name as an alphanumeric field.

* The only distinction between text and alphanumeric fields found in the Revelation manual is the following statement "R/LIST will break on spaces when printing a text field that is too large for the column"

PROGNAME	PROGRAM

PROGNAME in the relation represents the program name and the key

Fig 7.4 Program Storing Relation

The file PROGRAMS contains only binary programs compiled outside Revelation, namely in DOS. R/BASIC programs, which are internal Revelation programs, are compiled and their binary programs are CATALOG-ed to become part of the vocabulary. That is, their names are inserted into the VOC file. These R/BASIC programs then drive (through the PCPERFORM instruction) the binary programs in the file PROGRAMS to perform tasks which R/BASIC programs cannot do. Such tasks are mainly the graphics facilities. (Note that Revelation provides no graphics facilities of its own).

To clarify what has been described in the previous paragraph see figure 7.5, which is an example of an R/BASIC program, called VIEW, for displaying pictures on the screen. Typing VIEW at the Terminal Command Level(TCL) will invoke the Turbo Pascal program (from the file PROGRAMS) for displaying the pictures on the screen. The actual command is:

:VIEW *picture-name*

where *picture-name* is the name of the database picture to be displayed.

7.7 Picture Manipulation in the Database

The pictures are processed by the ordinary commands found in the data base management system. For example the following R/LIST query command will produce the picture called WORLD.MAP on the laser printer.

:LIST LASERPICS "WORLD.MAP" ID-SUPP COL-HDR-SUPP SUPP (P)

We only need to SUPPress default headers, footers, and the key to get the picture printed

properly.

When displaying the picture on the screen, however, the picture bytes have to be written into a DOS file. From this file, a program then displays the picture on the screen, see figure 7.5. This is necessary because Revelation does not provide graphics facilities.

```
************************************************
**** get file name from command line as second field
STRG = TRIM(@SENTENCE)
PICNAME= FIELD(STRG, ' ',2)
IF LEN(PICNAME) < 1 THEN
  PRINT ''
  PRINT "PICTURE FILENAME MISSING"
  PRINT "USAGE IS: VIEW picture-name"
  ABORT
END
**** open file of screen displayable pictures
OPEN '','SCREENPICS' TO FILE.PIC ELSE
  PRINT "CAN'T OPEN SCREENPICS"
  ABORT
END
**** extract named picture from file
READ PIC FROM FILE.PIC,PICNAME ELSE
  PRINT "PICTURE ":PICNAME: " NOT IN DATA BASE"
  ABORT
END
**** write picture bytes to DOS file PICTURE
OSWRITE PIC TO 'PICTURE'
****
**** open programs file
OPEN '','PROGRAMS' TO FILE.PRG ELSE
  PRINT "CAN'T OPEN FILE PROGRAMS"
  ABORT
END
**** extract dislaying program
READ PROG FROM FILE.PRG,"LOADWIN.COM" ELSE
  PRINT "PROGRAM LOADWIN.COM NOT IN DATA BASE"
  ABORT
END
**** write displaying program to DOS
OSWRITE PROG TO 'PROG.COM'
****
**** display picture on screen
PCPERFORM "PROG.COM PICTURE"
STOP
END
************************************************
```

Fig 7.5 Program VIEW for displaying pictures

In order to display a picture on the screen, (that is, after typing VIEW *picture-name* at the terminal command level) the following are performed inside program VIEW.

(i) select picture to be displayed from database

(ii) write selected picture bytes to a DOS file

(iii) select the displaying program from database

(iv) write displaying picture to a DOS file

(v) display picture through PCPERFORM instruction

The only limitation when selecting and extracting a picture or a program is that the whole record must be extracted. Extracting them as fields does not work because Revelation software truncates it when it encounters either character FE or FF(hexadecimal) because these are field and value markers. But these characters can occur in any TEXT field.

7.8 Performance Comparison

This section tries to monitor the performance of the picture manipulation functions described in the previous sections. Such a process is important because in most systems, there are always trade-offs to be made especially, between storage space and execution time. One has to sacrifice one for the other. In our case, the following tradeoffs will be considered:

(a) whether to store the bit-mapped pictures or

(b) the data the pictures were generated from.

These are to be weighed against the speed of getting the picture displayed on the screen.

For example for digitised coordinates, if the file of coordinates is smaller than the corresponding bit-mapped picture, then less space stores the coordinates than storing the actual picture. But a question arises, what about speed of getting the picture displayed? Similar questions can be asked for large coordinate files.

Two files of digitised maps used later as data in picture production and performance assessment need special mention here, these are:

(1) A file containing the coordinates of the world map. This file contains 18932 coordinates(vectors), and

(2) a file containing the coordinates of a triangle. This file contains 4 vectors.

Experiments (a) to (s) were done to measure performance in relation to use of storage space and CPU execution time in manipulating pictures. R/BASIC programs were executed in

Revelation in all the experiments. The experiments can be classified into several groups namely A, B, C, D, and E. Each of these is described below.

~~~~~~~~~~~~~~~~~~~

(A) Maps with a Large Number of Coordinates

Here we need to check the speed of generating against the speed of displaying a stored picture. The coordinates of the world map described earlier were taken as an example. The following cases were considered:

(a) generate map on entire screen with coordinates on floppy disk

(b) generate map on 1/4 of screen with coordinates on floppy disk

(c) generate map on entire screen with coordinates on hard disk

(d) generate map on 1/4 of screen with coordinates on hard disk

(e) display full screen map with image file on floppy disk

(f) display 1/4 screen map with image file on floppy disk

(g) display full screen map with image file on hard disk

(h) display 1/4 screen map with image file on hard disk

~~~~~~~~~~~~~~~~~~~

(B) Maps with Small Number of Coordinates

Here we need to check the speed of generating against the speed of displaying the resulting stored picture. The file containing the coordinates of a triangle, described earlier was used as an example. The following cases were considered:

(i) generate triangle on entire screen with coordinates on floppy disk

(j) generate triangle on 1/4 of screen with coordinates on floppy disk

(k) generate triangle on entire screen, coordinates on hard disk

(l) generate triangle on 1/4 of screen, coordinates on hard disk

(m) display picture of triangle onto screen

(C) Displaying Pictures from a Revelation Database

The aim here is to check speed of displaying pictures stored in data base. Because Revelation does not provide graphics facilities, pictures in the data base must be retrieved and put into a DOS file, then displayed from there. The DOS file into which the picture can be written can either be on hard disk, floppy disk, or virtual disk. (Note: a virtual disk in this case means an area in memory above the DOS limit of 640K which is referenced as a drive. The only difference between the virtual disk and a normal disk is that when there is a power lost everything on the virtual disk is erased). So the following were considered:

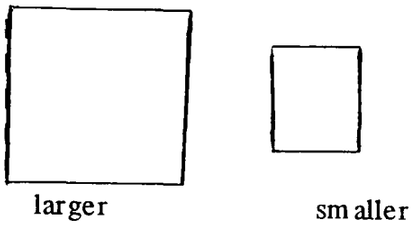
(n) Picture file put on hard disk

(o) Picture file put on floppy disk

(p) Picture file put on virtual disk

(D) Scaling Pictures by Constant Factor

The aim here is to measure the time taken to scale a bit-mapped picture stored in a file.



(q) Enlarging the Picture of World to Twice its size

If j is scale factor, the image expansion of a bit-mapped picture is done by duplicating each row and column j times.

(r) Shrinking the Picture of World to Half its size

If j is the compression factor, a simple compression algorithm is used which selects every j-th column and every j-th row.

(E) Retrieving pure text from a Revelation Database

The aim here is to check the speed of text retrieval from the data base and compare it with the speed of displaying pictures. This is important because, text retrieval is the only query processing facility provided in Revelation query language R/LIST. This test will give an indication of the speed of query processing in the database, therefore unabling us to say whether picture processing is fast enough.

(s) Retrieving 458 records from Revelation file.

There is no magic about the number 458 records, it only happens that it is the number of the records in one of files in the Revelation Demonstration kit. This kit contains a set of data

used as examples used when teaching or learning Revelation.

Experiment	Elapsed Time in secs. Buffers 15 trials 1	2	3	avg.	%err	2 trials 1	2	3	avg	%err
(a)	611	611	614	612	0.3	614	615	613	613	0.3
(b)	611	610	615	612	0.5					
(c)	450	456	453	453	0.7					
(d)	450	452	451	451	0.2					
(e)	14	15	15	15	6.7	15	14	15	15	6.7
(f)	7	7	7	7	0.0					
(g)	5	5	5	5	0.0					
(h)	4	4	4	4	0.0					
(i)	4	4	4	4	0.0	4	4	4	4	0.0
(j)	4	4	4	4	0.0					
(k)	3	3	3	3	0.0					
(l)	4	4	4	4	0.0					
(m)	5	5	5	5	0.0					
(n)	9	8	9	9	11.1	10	10	10	10	0.0
(o)	19	20	18	19	5.3	22	24	26	24	8.3
(p)	7	7	7	7	0.0	9	8	9	9	11.1
(q)	63	60	57	60	5.0	64	64	67	64	4.7
(r)	20	20	19	20	5.0	21	23	23	23	8.7
(s)	27	27	27	27	0.0					

Fig 7.6 Results Table

7.9 The Results

It is important to stress that the main objective of these measurement experiments was to monitor the performance of the picture processing function as implemented. The aim being to see how good or bad the system performs on this hardware. On the contrary, the aim was not to prove that flopyy or hard discs are slower than virtual discs.

The results(elapsed times) in figure 7.6 were obtained by the use TIME() function in the Revelation DBMS which gives the time of the day in seconds. Elapsed time was found by subtracting the time of the day immediately before the activity was started from the time of the day immediately after the activity was completed.

Average elapsed times are given in the column labelled avg. These average times, given to the nearest second, were obtained by averaging the elapsed times measured by repeating the experiment three times.

The column labelled %err in figure 7.6 gives the maximum error of the elapsed times from the average. As it can be seen, the difference in elapsed time from experiment to experiment did not differ much. The maximun difference is about 11.1 percent.

The measurements given when the number of buffers is two are for comparison purposes only. The aim was to see what happens when an unrecommended value for the number of buffers (the manual says between 10 and 20) is used.

7.10 Observations, Recommendations & Conclusions

7.10.1 Observations

(1) Picture operations from floppy disk take longer than (about 111.1 percent more) those on hard disk for this hardware.

(2) It is faster (about 28.6 percent more) to process pictures on virtual disk than on hard disk for this hardware.

(3) It takes about 7 seconds to display a picture from the data base onto the screen.

(4) It takes more time to draw a picture/map from many vectors than one from fewer vectors. This is also expected.

(5) Drawing speed does not depend on window size, see experiments (a) and (b), (i) and (j). These show that the speed of graphics package function to map world coordinates to window or screen coordinates does not depend on window size.

(6) For pictures generated from very few vectors, it is a bit faster to generate the picture than to display its stored image. In the triangle example, generation speed is faster by about 25 percent than displaying speed. However, for pictures generated from relatively many vectors, it takes much more time to generate the picture than to display its stored image. In the world map example, generation time is 8000 percent more than the time needed just to display the stored picture.

(7) In the DOS manual, it is recommended that for data base applications the number of buffers should be between 10 and 20. In the above experiments the number of buffers was initially set to 15. Some of the experiments were repeated when number of buffers is 2, the default. The results show that the values obtained are consistently higher when the number of buffers is two than when the number of buffers is fifteen.

7.10.2 Recommendations

(i) Pictures should be displayed from their stored bit-maps and not generated from

digitised map vectors by programs.

Although this will mean more storage space for pictures with relatively small number of vectors, this strategy will give a uniform view of how pictures are processed. Note: pictures from Microsoft Windows are not generated from vectors, but are drawn on the screen by use of a mouse. Furthermore, the loss in space is not critical since each picture file is less than 32K long. (Remember that the display processor used can generate pictures whose maximum size is 712 pixels by 350 pixels.)

(ii) To increase processing speed all temporary files should be written to virtual disk and not to the hard disk. This will improve processing speed by about 28.6 percent.

(iii) Pictures should be generated in sizes they are mostly likely to be used than scale them up or down later because scaling bit-mapped pictures is expensive in terms of CPU time, while during picture generation, generation time does not depend on window size.

(iv) The 7 seconds needed to display a stored picture on the screen though slow for purely interactive graphics applications[34, 53, 42], where almost instant response is required, is adequate for applications where retrieval of stored pictures is needed.

7.10.3 Conclusions

It has been shown that it is possible to store pictures in a relational data base. By treating the pictures in the data base as ordinary text fields they can be manipulated by the functions offered by the data base management system. It has further been shown that the performance of such a system is adequate for systems where retrieval of stored pictures is needed.

This storage and retrieval of pictures and binary programs has been implemented on an IBM-PC within the environment offered by the MS-DOS operating system. Transporting the overall system to other environments should therefore, in principle, not be too difficult. Indeed, porting it to other relational DBMS environments, (for example, INGRES) should not pose too many problems provided the target environment has a graphics package that follows the Graphics CORE Standard.

Porting it to an environment that does not follow the Graphics CORE Standard raises the question of the portability of this implementation from an environment based on the Graphics CORE Standard to a system based on, for example, the GKS environment. When porting to a host system that follows the GKS standard, two outcomes are possible. First, if the new system is programmed in pure GKS, then this will obviously not pose very many problems. Second, if the new system is not programmed in pure GKS but at some other level (note that graphics standards are defined in several levels) then this could pose severe problems and might entail rewriting the graphics routines.

Chapter 8

The Relational Multimedia Data Base System

8.1 Specification

8.1.1 The Making of RMDBS

Following the successful demonstration of storing and processing bit-mapped pictures in a traditional relational DBMS (as discussed in the previous chapter), and the adequate performance of the picture processing functions, a system that takes advantage of such success, the Relational Multimedia Database System, in short RMDBS, has been designed and implemented. RMDBS is first and foremost a system to efficiently store and manipulate various kinds of data: numeric and text data, bit-mapped pictures, and programs in binary code.

RMDBS is an integrated system which enables the user to manage and control operations on text and numeric data, bit-mapped pictures, and binary programs wholly within the system. The system is completely menu driven, so the user does not need to remember any command names or their formats to work with the system.

8.1.2 Need for Picture Generating Capabilities in RMDBS

RMDBS is implemented in a normal relational DBMS which enables a user to work with the usual commands found in the DBMS, plus a few others for picture processing. Therefore:

(i) using the DBMS facilities an RMDBS user can select data from the database files satisfying certain selection criteria, or having certain properties;

(ii) the user might further require that the data selected be used to generate a picture. To be able to support the latter requirement, RMDBS therefore must have some picture generating capabilities.

If RMDBS is to support data selection from the database, and getting its pictorial representation without some picture generating capabilities, then every picture corresponding to

every conceivable combination of selection criteria, must be stored in the database. It is obvious that this is almost impossible to accomplish in most computer systems because of the limited storage space available on the disk. For example, suppose were are working with maps of counties in the UK. Without some map generating in the system we need to generate all maps and all their combinations. If all combinations of county maps in the UK are to be stored, and since there are 66 counties, it follows that 66! = 66x65x64 ... 3x2 pictures need to be kept on the disk. This is a very large number. Even if each picture was only one byte long, such a large amount of storage space is not available even on most large computer systems.

The only viable solution is therefore to include picture generating capabilities in RMDBS. This will enable the user to select data, and use it as source of a picture plot if he/she chooses to. Alternatively, the generated picture can be stored in the database, printed to get a hard copy or just ignored. Only a few pictures therefore need to kept in the database. These will be pictures which are stored because generating them is time consuming or they are very frequently used, see section 7.10. This way efficient use of disk space can be maintained and system processing speed optimised.

8.1.3 A Demonstration System for RMDBS

A demonstration system for RMDBS has been implemented on data pertaining to counties in the UK. Two types of data used are: (i) statistical data, like county populations, etc. (this data is stored in relations and can be selected and used to generate business or statistical charts); and (ii) digitised coordinate (x,y) pairs for each county. (this data can be selected and used to plot county maps).

8.2 System Overview

Below is the main menu a user sees when he/she logs into RMDBS. The list shows the different functions offered by RMDBS and their choice numbers. A detailed description of each function is given later.

NO. FUNCTION

00 Get HELP Information

01 VIEW: display a screen database picture onto screen

02 PUT.SCRPIC: insert a screen picture into database

03 PUT.LSRPIC: insert a laser picture into database

04 PUT.PROG: insert a binary program into database

05 HARD.COPY: print a laser printable picture

06 PRINT: a screen picture onto laser printer

07 SCALE: scale a picture up or down by constant factor

08 OVERLAY: merge two pictures to produce a third picture in a DOS file

09 SMART: execute Smart Spreadsheet to draw/edit a picture

10 WIN: execute Windows Paint to draw/edit a picture

11 DRAWON: retrieve data and draw it on top of a picture

12 MAIN.MENU: perform the main Revelation MENU

13 DEL.PICS: delete 'least used' pictures from database

14 CHANGE.PARS: change system parameters

15 SEE: what programs, screen or laser pictures are in the database

16 DELETE: a program, screen or laser picture from database

17 EXIT to TCL

As mentioned earlier, RMDBS is a completely menu driven system. The user only enters the number of the function he/she wants and the system moves to the next stage. Even in stand alone subsystems used in RMDBS like Smart, automatic methods of selecting and executing functions have been implemented in order to reduce the amount of command memorisation required by the user. Help information is also provided at every stage. It is hoped these measures go a long way in making RMDBS user friendly.

How RMDBS is invoked

The RMDBS is invoked by first logging into the Revelation DBMS and typing the following at the Terminal Command Level.

:RMDBS

Before the functions provided by the system are described in detail, the relations or files used in the system are first described in the next section.

8.3 Relations Used

This section looks at the relations used to store, manipulate, and manage the pictures (both screen and laser printable pictures), binary programs, xy coordinate vectors(for generating maps), and other numeric and text data.

8.3.1 Relations for Screen Pictures

Two relations are used to store, manipulate, and control operations on screen displayable pictures in RMDBS. These are SCREENPICS and PIC.ATTRS. Figure 8.1 illustrates their schemas.

SCREENPICS
(a)

PICNAME	PICTURE

PIC.ATTRS
(b)

PICNAME	DATE.PUT.DB	LAST.DATE.USED	SOURCE.PROG	DESC

Fig 8.1 Relations for Screen Pictures

The relation in figure 8.1(a) stores the actual screen pictures. It has only two domains; PICNAME and PICTURE. PICNAME, the name of the picture, is the key. The bit-mapped picture itself is stored in the PICTURE field as a long text string(about 32K bytes long).

The relation in figure 8.1(b) keeps the picture characteristics. The DESC field is a one line description of the picture. It is given by the user when the picture is being inserted into

the database. It helps users know what the picture is about.

The relation in figure 8.1(b) is used in "deciding" what pictures are to be deleted from the database when their number exceeds the recommended maximum number of pictures. Deleting pictures from the database is necessary because disk space is finite, so with time and more pictures being stored in the database, the disk will fill up.

8.3.2 Relation for Laser Printable Pictures

Laser printable pictures are stored in the relation LASERPICS, whose schema is shown in figure 8.2. The relation has two fields only. The field PICNAME, the name of the picture, is the key and the laser printable picture is stored in the next field. This field is a long text field containing the picture bits and the control characters (put in by the formatter) needed to drive the printer.

LASERPICS

PICNAME	PICTURE

Fig 8.2 Relation for Laser Printable Pictures

To conserve disk space, it is recommended that very few pictures should be relation LASERPICS because if the user wants a hard copy of a picture, the equivalent screen picture can be be printed easily. But the function is there for the user to use if he/she wants.

8.3.3 Relations for Binary Programs

To store and to control operations on the binary programs used in the system, two relations, whose schema are shown in figure 8.3 are employed. The programs themselves are stored in the relation PROGRAMS and their characteristics in PROG.ATTRS.

The DESC field is a one line description of the program. It is given by the user when the picture is being inserted into the database. It helps users know what the program is all about.

PROGRAMS
(a)

PROGNAME	PROGRAM

PROG.ATTRS
(b)

PROGNAME	DATE.PUT.DB	LAST.DATE.USED	SOURCE.LANG	DESC

Fig 8.3 Relations for Binary Programs

The programs stored in the relation PROGRAMS are binary programs compiled outside Revelation. The source language of most of these is Turbo Pascal, but there are a few 'C' and assembly programs.

It is recommended that programs should not be deleted from the database just to get more space on the disk, because if a picture is (accidentally) destroyed, and if the generating program is not there then there is no way that the picture can be generated (unless of course one writes or recompiles the original program). This of course does not stop the user from deleting unwanted programs.

8.3.4 The Parameters Relation

To make the system more flexible to changes in parameter values, (for example: a change in MAX.NO.PICS, the maximum number of pictures allowed in database, or type of printer,

etc.) a separate parameters file, PARAMS.FILE is maintained. The schema of this relation is represented in figure 8.4.

PARAMS.FILE

PARAM.ID	PARAM.VAL

Fig 8.4 The Parameters Relation

Each parameter is given an id or name. This name (which is also the key) is stored in the first field. In the second field the value of the parameter is stored.

All programs in the system refer to this file for their parameter values. This arrangement allows easy changes to parameter values because only one program to enter or update parameters is needed for the whole system.

8.3.5 Relation for Digitised Maps

The Relation SHAPES, figure 8.5 below, contains the xy coordinate vectors for generating the county maps of counties in Great Britain. Data in this relation were obtained from the GIMMS package. Each map is represented by two fields, SHAPE.ID (which is the key) and the multi-valued field (X, Y, FLAG). This multi-valued field repeats itself as many times as the number of coordinates in the map.

Note

A FLAG is just a character associated with each digitised point. By digitising with different buttons on the cursor pad, a varying FLAG may be stored with each digitised point. Commonly, flags are used to signal either a pen or colour change, a change from 'join' to 'move' in the plotting sequence, etc.

SHAPES

SHAPE.ID	X	Y	FLAG
	x1	y1	f1
	x2	x2	f2
		...	
	xn	yn	fn

Example entries from relation SHAPES are:

TRIANGLE	330	300	1
	330	300	0
	5000	5000	0
	8981	300	0
	330	300	1
LINE	678	800	1
	678	800	0
	6000	6000	0

Fig 8.5 Relation for Storing Digitised Maps

SHAPES is the only relation used in this system that is unnormalised, because it consists of repeating fields. This property of employing unnormalised or ordered tuples in relational databases is essential when pictures are to be generated from pictorial-based data stored in relations, see also[39, 34, 62].

8.3.6 Relation for Census Data

The COUNTIES relation contains county data from GIMMS and SASPAC packages. Its schema is represented in figure 8.8.

COUNTIES

NAME	SHAPE.ID	X	Y	POP	SIZE	BORN.ENG

Fig 8.6 The COUNTIES Relation

(X,Y)

Is the centroid position of the map. This position is good for labelling maps On ordinance survey maps, X is the easterly distance from the west most line, whereas Y is the northerly distance from the most south most line. Both X and Y are given in metres.

POP

Is the county population in thousands.

SIZE

Is the size of the county in thousand square metres.

BORN.ENG

Is the number of people in the county born in England.

This same data is also stored in the DOS file, CENSUS, from which it can be accessed and used in the Smart Spreadsheet program as a worksheet.

Together with the relation SHAPES one can select data from COUNTIES and/or SHAPES through the R/LIST query language, then either:

(i) plot county maps

(ii) write county names at position (x,y)

(iii) put the data in a worksheet to generate charts/diagrams in the Smart Spreadsheet.

For example one might like to:

(a) Plot the maps of counties with X > 56,000 that is, those counties whose centres are 56,000

metres from the left most point on the ordinance survey maps; or (b) generate a pie chart of county populations for counties with size over two million square metres.

Note

Although the actual selection of data from the database is done in R/LIST, in this system the user does not need to know the actual syntax of R/LIST commands. He/she needs only to enter the selection criteria as a boolean expression. The rest of the formulation, that is, to put the query in proper R/LIST syntax is done in the system.

8.4 Writing of Boolean Selection Expressions

The discussion below explains in some detail how the boolean expression is entered by the naive user. (The experienced R/LIST user is given the option to go straight to writing the boolean expression in R/LIST syntax).

The method used is to allow the user to write the boolean expression in a straight forward but structured way. This expression is then converted into R/LIST in the system.

Using BNF notation, a boolean expression is defined as:

< boolean expression> := < boolean expression> < connector> < token> |< token>

where;

< token> := < variable> < operator> < value>

< variable> := NAME |X |Y |SIZE |POP |BORN.ENG

< operator> := < |> |< > |= |< = |> =

< connector> := OR |AND

Using the definitions of < variable> , < operator> , and < connector> given above, a simple parser was written. This parser is able to check that the expression entered conforms to this syntax.

If it conforms, the expression is then put into the correct R/LIST syntax.

If it does not conform, the user is told so and requested to re-enter the expression.

In the above definition, < value> is allowed to take anything, number or string. The system requires a value to be entered, if it is not given the expression is incomplete, so it is rejected. The user is then asked to re-enter the expression.

When the variable is NAME, the < value> entered is quoted to conform with R/LIST syntax for string values.

Two schemes are available to the user when entering the boolean expression either (a) enter the expression one token at a time or (b) enter the whole expression.

8.5 Description of Functions

This section describes in detail the functions listed in section 8.2. But before that, two general comments need to be made here. These are:

Comment 1

Whenever a screen picture or program is referenced in any function, the LAST.DATE.USED field in the database relation PIC.ATTRS or PROG.ATTRS, respectively is updated to reflect that it has been used on a given date.

Comment 2

In all the functions where a new picture is generated, the user will see the following menu:

No.	FUNCTION
1	PRINT it on laser printer
2	STORE it into database
3	BOTH 1 & 2 of the above
4	EXIT: do nothing

This menu allows the user to decide what he/she wants to do with the new picture produced.

8.5.1 Get Help

Choice No: 00

Function: Get Help Information

Explanation:

Gives help information at main system level. The help information explains to the user what the system does, how to make choices, etc. The user is recommended to read this information. Help information will be given at every important stage of the system.

8.5.2 Display Picture

Choice No: 01

Function: Display a picture on the screen

Explanation:

Selects a picture from the database and displays the picture on the screen.

User action:

The user will be requested to enter the name of the picture to be displayed.

8.5.3 Insert Screen Picture

Choice No: 02

Function: Insert a screen picture into database file

Explanation:

Puts a screen displayable picture from a DOS file into the database file SCREENPICS.

If the number of pictures in the database equals or exceeds MAX.NO.PICS, the program DEL.PICS (see choice no 13) is called to delete "least used" pictures from the database to give room on the disk, before the picture is stored in the database.

User action:

The user will be requested to enter the following:

1) name of the picture

2) full path name of the DOS file containing the picture

3) source program, that is, the program that generated the picture. The source program can either be Windows, Smart or Turbo Ppascal

4) a one line description of the picture

8.5.4 Insert a Laser Picture

Choice No: 03

Function: Insert a laser printable picture into database

Explanation:

Puts a laser printable picture from a DOS file into the database file LASERPICS.

User action:

The user will be requested to enter the following:

1) name of the laser picture

2) full path name of the DOS file containing the laser picture

8.5.5 Insert a Program

Choice No: 04

Function: Insert a binary program into database file

Explanation:

Puts a binary program from a DOS file into the database file PROGRAMS.

User action:

The user will be requested to enter the following:

1) name of the program

2) full path name of the DOS file containing the program

3) source language, that is, the type of source language. The source language can either be Turbo Pascal, 'C', or assembly.

4) a one line description of the program

8.5.6 Print a Laser Picture

Choice No: 05

Function: Print a laser printable picture

Explanation:

Selects a laser printable picture from file LASERPICS, and prints it onto the laser printer.

User action:

The user will be requested to supply the name of the picture to be printed.

8.5.7 Print a Screen Picture

Choice No: 06

Function: Get a hard copy of a screen picture

Explanation:

Takes a screen displayable picture from database, laser dumps it, and prints it onto the

laser printer.

User action:

The user will be requested to supply the name of the screen picture to be printed.

8.5.8 Scale a Picture

Choice No: 07

Function: Scale a screen picture

Explanation:

Scales a screen picture from the database up or down by a constant factor.

User action:

The user will be required to choose the type of scaling. Whether to scale:

1) up

2) down

After selecting the function, the user will then have to enter the scaling factor.

8.5.9 Overlay two Pictures

Choice No: 08

Function: Overlay two screen pictures to produce a third in a DOS file

Explanation:

Combines the pixels of two database screen pictures to produce a third picture. Combining is done by ORing corresponding pixels in the two pictures. The new picture is left in a DOS file. (Note: this algorithm is fine for monochrome pictures. For coloured pictures this needs to be changed otherwise random colours will be produced.)

User action:

The user will be required to supply the names of the two database screen pictures to be combined and the name of the new picture.

8.5.10 Run Smart

Choice No: 09

Function: Execute the Smart Spreadsheet

Explanation:

Executes the Smart Spreadsheet either to draw a new picture or to edit an old picture. A new Menu is displayed:

No.	Function
1	Get HELP information
2	DRAW a new picture
3	EDIT an old picture

8.5.10.1 Help

Choice No: 1

Function: Get Help information

Explanation:

Explains to the user how and what can be done in Smart and how user can form a query to select data from the database for use in Smart.

8.5.10.2 Draw Picture

Choice No: 2

Function: Draw a new picture

Explanation:

Allows the user to execute the Smart Spreadsheet in order to define and generate a new picture. The full facilities of the Smart program are at the disposal of the user. This enables him/her to define and generate pictures as desired. Two options are open to the user:

1) use any worksheet to define and generate a new picture

2) retrieve data from the database and use this data to generate a picture

8.5.10.2.1 Generate Picture from any Worksheet

Choice No: 1

Function: Use any worksheet to generate picture

Explanation:

Executes the Smart Spreadsheet to generate a new picture. Any worksheet, for example CENSUS, can be loaded and used to define and generate a picture.

Since the Smart Spreadsheet gives the user an empty screen, some way of automating these processes is desirable. The use of "projects" is very useful in this case. Using projects it is possible to give the user some help information and to automatically choose and execute the functions he/she wants.

Through the use of projects, figure 8.7 is the message that appears on the screen to prompt or help the user, especially the non-experienced Smart user.

You are now in the Smart Spreadsheet program, ready to Define and Generate a Screen picture.

Note the following:

1. The worksheet is already loaded in memory

2. While defining the picture, follow the menus shown on the picture definition screens. But note the following, since they are not clearly explained in Smart:

   i) Press F6 to access data in the worksheet
   ii) Use the arrow keys to move cursor to the start of the block of data you are interested in
   iii) Press F2 to signal to the system that this is the start of the data block. In Smart this is known as to "drop anchor".
   iv) Use the arrow keys again to move the cursor, but this time to the end of the data block. If done correctly, the block of data marked will be reverse video.
   v) Press the RETURN key to go back to the picture definition screen

3. The picture will be automatically GENERATED after its Definition.
   REMEMBER to save the screen picture, you will be prompted for a name!
   After this Smart will be exited.

Fig 8.7 Help Information Automatically Displayed in Smart

The above is made possible by use of two projects: The first one, figure 8.8(a), is executed in the Smart Word-processor to display the help information in figure 8.8. The second one, shown in figure 8.8(b) is executed in the Spreadsheet. This enables automatic:

(a) loading of the worksheet

(b) start of the picture definition process

(c) generation of the screen picture

```
read helper
MESSAGE Press ANY key to CONTINUE
JUMP Spreadsheet PROJECT-FILE auto
```

(a)

```
%1 Enter the name of the worksheet
@r1c1 read text %1
%2 Please ENTER name of Picture
GRAPHICS define %2
GRAPHICS generate %2 black/white screen
QUIT
```

(b)

Fig 8.8 Projects to Define/Generate Pictures Automatically

User Action:

By use of the projects shown in figure 8.8, user action is reduced to only filling in the picture attributes (for example name, headings, footers, labels, etc) in the picture definition screens.

8.5.10.2.2 Generate Picture from data Selected from Database

Choice No: 2

Function: Select data from the database to define and generate a new picture

Explanation:

Allows the user to select data from the relation COUNTIES, and use this data to generate a new picture. The data is retrieved and put in the worksheet \TMP\DATAFROM.REV. The user then uses the Smart Spreadsheet to define and generate the new picture. In Smart the worksheet is accessible only as DATAFROM.REV.

This is a very interesting part of the system, because it allows the user to select data satisfying certain criteria and use this data only to draw a picture.
For example the user might want a histogram of populations of counties whose sizes are greater than ten million square metres.

In such a case the user will only need to type the selection criteria:

SIZE > 10000

Records in the COUNTIES relation satisfying this selection criteria will be retrieved and put in the worksheet \TMP\DATAFROM.REV, and from here used in the Smart Spreadsheet to define and generate a screen picture.

Limited help to the user, in the form of examples of how to write the boolean expressions, is available on-line.

Again projects are used to reduce the number of commands the user has to memorise in order to accomplish the needed tasks. The same help information, shown in figure 8.7, is displayed on the screen. Then the project in figure 8.9 is used to load the worksheet, automatically start the picture definition process, and finally to automatically generate the picture.

```
@r1c1 read text DATAFROM.REV
%1 Please ENTER name of Picture
GRAPHICS define %1
GRAPHICS generate %1 black/white screen
QUIT
```

Fig 8.9 Project to Define/Generate Pictures Automatically

User action:

Through the use of the project in figure 8.9, user action is merely to:

(a) enter the selection criteria of data to be retrieved

(b) respond to the prompts in the picture definition screens.

Comment

Of the two methods for generating a new picture in Smart, described above, method two is recommended where data satisfying given conditions is required to generate the picture. This is because, the data in the worksheet DATAFROM.REV will already be satisfying the conditions.

For example if the user wants a histogram of populations of counties whose sizes are greater than ten million square metres, then the population field in the worksheet DATAFROM.REV qualifies the conditions, so definition and generation of the picture can go on without problems.

To use the worksheet CENSUS in such a case means that the user will have to check manually whether the record satisfies the selection condition. A cumbersome process prone to errors, especially if the selection criteria is a complex boolean expression. However, if the data does not need to satisfy any conditions, then method one, whereby the worksheet CENSUS is used is fine.

8.5.10.3 Edit a Picture

Choice No: 3

Function: Select a database screen picture and edit it as desired

Explanation:

Retrieves a named picture from the database and allows the user to edit it as desired in the Smart Spreadsheet.

The picture will be retrieved from the database and written to a DOS file, with the same name plus the extension ".SCN", in the directory \TMP, the home directory of Smart.

Following the same principle of reducing memorisation of the commands needed to edit a picture (just as for picture Definition and Generation), the project shown in figure 8.10 is used. Entering the Spreadsheet through this project makes it possible to automatically enter the picture editing command without going through the several menus the user has to choose.

```
@r1c1 read text helper3
MESSAGE Press ANY key to CONTINUE
%1 Please ENTER name of Picture
GRAPHICS edit %1
QUIT
```

Fig 8.10 Project to Start Picture Editing Automatically

User action:

Again by use of a project, user action has been cut to:

(i) entering the name of the picture to be edited

(ii) editing the picture as desired

8.5.11 Run Microsoft Windows

Choice No: 10

Function: Execute the Windows Paint program to draw or edit a picture

Explanation:

Allows the user to draw a new picture in Windows or select a picture from the database and edit it in Windows.

A new Menu is displayed:

1) get help information

2) draw a new picture

3) edit a database picture

8.5.11.1 Help

Choice No: 1

Function: Get Help information

Explanation:

Explains to the user the power of the Windows Paint program to draw and edit pictures.

8.5.11.2 Draw Picture

Choice No: 2

Function: Draw a new picture

Explanation:

Executes Microsoft Windows, giving the user the full power of the system to draw a new picture. To be able to execute the PAINT.EXE program, the user will have to change directory to \WINDOWS, the home directory of Windows.

IMPORTANT

while in the PAINT program make sure the option set for pictures is TERMINAL and not for PRINTER

User action:

To draw a new picture in Windows the user needs to do the following:

(a) change directory to \WINDOWS

(b) execute the PAINT.EXE program

(c) select the Option TERMINAL (this is very important, otherwise the picture drawn will not be compatible with other system functions)

(d) draw the picture

(e) save the picture in a file

(f) quit Windows

8.5.11.3 Edit a Picture

Choice No: 3

Function: Select a database screen picture and edit it as desired

Explanation:

Allows the user to select a named screen picture from the database and edit it as desired.

The picture will be retrieved from the database and written to a DOS file, with the same name plus the extension ".MSP", in the directory \WINDOWS, the home directory of Windows.

User action:

To edit the picture in Windows the user needs to do the following:

(a) change directory to \WINDOWS

(b) execute the PAINT.EXE program

(c) open and load picture to be edited

(d) edit the picture as desired

(e) save the picture in a file

(f) quit Windows

8.5.12 Draw on a Picture

Choice No: 11

Function: Retrieve data from database and draw it on top of a database picture

Explanation:

This function allows the user to select a screen picture from the database, select data from the database and draw it on top of the selected picture.

This function, therefore, allows the user to start with a picture or map (which can be blank) and progressively add more details to it.

The data is selected from two relations; COUNTIES and SHAPES.

Maps of selected counties or/and their names can be drawn on the picture. A new Menu is displayed:

1) draw county maps

2) plot county names at centroids of their maps

3) draw & label county maps

4) help information

User action:

In both situations, the user must supply the following:

(a) the name of the database picture to draw on

(b) the selection criteria of the data to be retrieved

To enter the selection criteria, the user only needs to type in the boolean expression of the selection condition.

For example, if the maps of counties with x satisfying x > 56,000 are to be plotted, then the user needs to enter the selection criteria:

X > 56000

On-line help information exists in the form of examples of how to write the boolean selection expressions. The user can choose to see this help information.

8.5.13 Descend to Main Revelation Menu

Choice No: 12

Function: Descend to the main Revelation Menu

Explanation:

This function allows a user to leave RMDBS and descend to the main Revelation Menu. This is needed when the user wants to access R/DESIGN to create other applications.

8.5.14 Delete "least used" Pictures

Choice No: 13

Function: Delete database screen pictures to give room on the disk

Explanation:

This function deletes "least used" pictures from the database to give room on the disk. This is analogous to an operating system replacing a page, simply because it has been resident in main memory longest!

The function uses two parameters from the PARAMS.FILE, these are:

NO.PICS - this is the recommended number of pictures to be held in the database

MAX.NO.PICS - this is the maximum number of pictures allowed in the database

When this function is chosen, a of count how many pictures are currently in the database is made:

(a) if the number of pictures in the database is less than NO.PICS, then the program just exits, nothing is done.

(b) if the number of pictures in the database is greater than NO.PICS, then the (MAX.NO.PICS - NO.PICS) pictures with the least LAST.DATE.USED are deleted from the database.

8.5.15 Set/ Update Parameters

Choice No: 14

Function: Change system parameters

Explanation:

This function allows the user to change or enter new values of parameters in the PARAMS.FILE.

User action:

For each new parameter and for each of the parameters whose values are to be changed, the user enters the:

(i) parameter name or id

(ii) parameter value

8.5.16 See names of Programs & Pictures

Choice No: 15

Function: See the names of programs or names of pictures stored in the database

Explanation:

This function allows the user to see what programs, laser pictures, or screen pictures are in the database. A new menu is given.

No	Function
1	see names of PROGRAMS

2 see names of LASER pictures

3 see names of SCREEN pictures

8.5.16.1 Names of Programs

Choice No: 1

Function: See the names of binary programs in the database

Explanation:

Lists on the screen the names of programs in the database.

User action:

Press the RETURN key after every screen full and to return to main menu.

8.5.16.2 Names of Laser Pictures

Choice No: 2

Function: See the names of laser printable pictures stored in the database

Explanation:

Lists on the screen the names of laser pictures in the database.

User action:

Press the RETURN key after every screen full and to return to main menu.

8.5.16.3 Names of Screen Pictures

Choice No: 3

Function: See the names of screen pictures in the database

Explanation:

Lists on the screen the names of screen pictures in the database.

User action:

Press the RETURN key after every screen full and to return to main menu.

8.5.17 Delete Picture or Program

Choice No: 16

Function: Delete a program or picture from database

Explanation:

This function allows the user to delete unwanted programs, screen pictures, or laser pictures from the database.

This is the preferred method of removing unwanted pictures and programs from the database. A new menu is given:

No	Function
1	delete a PROGRAM
2	delete a LASER picture
3	delete a SCREEN picture

8.5.18.1 Delete Program

Choice No: 1

Function: Delete an unwanted program from the database

User action:

Enter the name of the program to be deleted from the database.

8.5.18.2 Delete Laser Picture

Choice No: 2

Function: Delete an unwanted laser picture from the database

User action:

Enter the name of the laser picture to be deleted from the database.

8.5.18.3 Delete Screen Picture

Choice No: 3

Function: Delete an unwanted screen picture from the database

User action:

Enter the name of the screen picture to be deleted from the database.

8.5.18 Descend to TCL

Choice No: 17

Function: Exit RMDBS and descend to TCL

Explanation:

Leaves RMDBS and drops to the Terminal Command Level.

8.6 Conclusion

This chapter has described the work undertaken during the development of a retational multimedia data base system called RMDBS, written the environment and host facilities offered the Revelation DBMS. Detailed descriptions of the RMDBS end-user command set have been

given along with an outline of how these commands have been implemented. Particular emphasis has been given to the provision of graphics facilities which compensate for the lack of such resources within the host environment provided by the Revelation system.

By means of timing experiments, (discussed in the previous chapter sections 7.8 to 7.10), it has been shown that the performance of such a system is adequate.

Chapter 9

Conclusions & Future Work

9.1 Conclusions

An implementation of a multimedia database in a traditional relational DBMS has been presented. The database integrates the storage and manipulation of bit-mapped pictures, binary programs, and attribute data. The implementation aimed: (1) to make efficient use of storage space; and (2) to have a user friendly interface. To varying extents each of these objectives have been realised.

Chapters 5 and 6 of this thesis have brought together and analysed related work on the design of multimedia databases. Several powerful and elegant systems have been studied; these reflect the transition from traditional relational DBMS facilities, through systems for storing pictorial-data to fully fledged multimedia databases. Each of these systems, though good and powerful, have limitations. First, the multimedia databases that have been described previously are not true DBMS systems. This is because they do not follow the agreed principles of DBMS design and implementation - for example, they store data in conventional files. Second, the DBMS systems used for storing pictorial-data are not multimedia databases because they store attribute data only. No pictures, images or voice data are stored in the database. Although these multimedia systems provide functions to draw pictures and to manipulate them on the display, the pictures themselves are not stored in the database.

Since pictures in databases for storing pictorial-data can generated and displayed but not stored, users of such systems have no way of keeping images for later use. If a picture is needed later, then the user has to go through the generation process again. This can be very cumbersome especially if the picture generation process is complicated.

Although it is not wrong to implement stand alone multimedia databases, such an implementation misses the services offered by DBMS. Such services include concurrency control, access control, redundancy control, and indexing facilities. A stand alone data base

would, therefore, need these services to be provided for in its implementation. Since such facilities can be obtained from DBMS, providing them means wastage of resources which could be used otherwise. The consequence of this is that such systems take much longer to perfect.

The work described in this thesis is therefore quite novel in that a true multimedia database has been implemented within the framework of a traditional relational DBMS.

The main objective of this research, therefore, was to show that a multimedia database system can be designed and implemented using a traditional DBMS facility. This work starts in Chapter 7 where it is demonstrated that bit-mapped pictures and binary programs can be stored, retrieved and manipulated in a relational DBMS. The work continues in Chapter 8 where a multimedia database system is designed and implemented. The demonstration system, called Relational Multimedia Data Base System, in short RMDBS, is described. A detailed description of the RMDBS end-user command set together with how these commands have been implemented is given.

As well as offering the normal functions provided by a relational DBMS, RMDBS also provides functions to manipulate a picture in the following ways: (i) to display it on the screen, (ii) to print it on a laser printer, (iii) to scale it, (iv) to edit it as desired, (v) draw other details to it. Other facilities offered by RMDBS are to generate pictures from various sources, and to combine two pictures to generate a third picture.

By means of a series of timing experiments it has also been shown that the performance of such a system in terms of processing speed is quite adequate.

The RMDBS system has been implemented on an IBM-PC within the environment offered by the MS-DOS operating system. Transporting the overall system to other environments should therefore, in principle, not be too difficult. Indeed, porting RMDBS to other relational DBMS environments, (for example, INGRES) should not pose too many problems provided the target environment has a graphics package that follows the Graphics CORE Standard.

Recently, there have been movements from the *de facto* CORE standard to the sanctioned GKS standard. This raises the question of the portability of RMDBS from an environment based on the Graphics CORE Standard to a system based on the GKS environment. When porting RMDBS to a host system that follows the GKS standard, two outcomes are possible. First, if the new system is programmed in pure GKS, then this will obviously not pose very many problems. Second, if the new system is not programmed in pure GKS but at some other level (note that graphics standards are defined in several levels) then this could pose severe problems and might entail rewriting the graphics routines.

In implementing the database, (i) unnormalised relations (with ordered/multivalued tuples), and (ii) variable length records have been used. It is important to emphasise that these are not properties of the relational database model. However, as other researchers have discovered, use of these approximations is inevitable if unformatted data (text, images) are to be stored and manipulated in a relational DBMS[33, 71].

9.2 The User Interface and Its Limitations

It has been said that RMDBS is user friendly. This claim is only true in the sense that an inexperienced user can safely and easily work with the system. This has been achieved by making RMDBS menu driven.

No rigorous measurements or experiments were done to monitor whether the RMDBS user interface is truly user friendly in terms of human factors, as detailed by Shneiderman in[82], and Rubinstein and Hersh in[83]. For example, how to reduce human error rate, optimising functionality, making the interface as easy to use as possible, design of the screen layout etc.

A systematic study of human factors was not undertaken because: (i) of lack of time, and (ii) the study of human factors not being a major requirement of this research. However, two people worked with RMDBS during its implementation stage. This was an informal exercise, the aim of which was to discover any problems, omissions, unclear messages, and to see whether the system could be used by a non-expert user without extensive use of the user

manual. Because this evaluation procedure was an ad-hoc effort no formal record was kept of their interaction with the system and evaluation comments was made. However, their informal comments were very valuable and were used to modify the initial design. The final system, therefore, incorporates their comments and suggestions.

As the complexity of user interfaces increases[84, 85], it becomes necessary to apply some methodology and discipline to their programming. This is particularly true in the case of user interfaces to multimedia databases. The user interfaces to such systems are very complex and require much thought to design and implement them (this has been amply illustrated by the systems discussed in chapters 5 and 6). A systematic approach to solving this problem is therefore required.

One potential solution to this problem lies in the design and fabrication of special purpose software packages that support human-computer interaction. Typically, they are called User Interface Management Systems(UIMS). The benefits from the use of UIMS include reduced costs of constructing interfaces, smoother interaction, higher reliability, and more importantly the increase in interface consistency across applications. Knowledge can be gathered from the multimedia databases already in place to start work now on notation and functionality to build such systems.

It is important to emphasise that no use was made of an explicit user interface management system within this work. Instead, this function was provided implicitly by means of the facilities offered by the operating system of the host microcomputer that was used to implement the RMDBS. The omission of any consideration for the design and fabrication of a UIMS can be justified on three grounds: (1) it was not deemed to be a major requirement of the research; (2) the complexity of the interfaces in this work did not warrant the use of this approach, and (3) this is a rapidly expanding and developing area in which there are few rules and guidelines to aid the practitioner.

9.3 Thoughts on Future Work

To make RMDBS into a fully fledged multimedia database, extensions in the areas outlined below need to be made.

9.3.1 Basic Extensions

(a) Text Storage and Retrieval

The addition of facilities to enable the storage and retrieval of text (documents, letters, etc) is an important requirement within many application areas. Extensions in this area would therefore allow search and retrieval of the text for its contents. For example, this will allow the user to search for documents which deal with "houses" or "buildings" and so on. Retrieval of individual documents could be implemented either by (i) the classical informational retrieval method of keyword searches, or (ii) through document signatures. Details on document signatures can be found in[86].

(b) Picture Searching Techniques

Ideally, RMDBS should include some form of pattern recognition routines in order to search for pictures having particular characteristics. Facilities for searching and extracting patterns or structures from the database will be required. For example, the user might want to extract from the database pictures that contain image segments that resemble a car. This poses problems relating to defining and extracting patterns (or templates) from images. Note that what can be a natural pattern for the human eye/ear is not as easy to pin down in terms of computer-oriented patterns. For example, will the user be able to specify precisely the picture content of pictures that he/she wants?

Since picture (or voice) patterns are not easy to define they often require complicated and time consuming pattern recognition techniques. Two methods to reduce time in matching patterns could be considered:

(i) to require only a certain percentage of match, say 70 percent. This brings other issues like:

who decides on the matching percentage, the user or the system designer? will this really reduce matching time? etc.

While reducing the matching percentage might reduce searching time, there arise the problems associated with recall, that is, more material will be retrieved than desired.

(ii) the use of array processors. This will obviously increase processing power, but it brings other problems related to programming multi-processor machines.

(c) Facilities for Handling Sound

It is important that a multimedia data base contains facilities to cater for the storage of sound. This will obviously need specialised equipment. Furthermore, decisions would also have to be made on how voice data (or other forms of sound) are to be stored in the database. Two methods exist to represent sound: (i) as analogue signals, or (ii) as digitised data.

However, with current technology, complete extraction of information from digitised voice poses the same difficulties as those of matching for patterns in pictures or images as discussed in (b) above.

However, as is the case with pictures, sound messages could easily be archived in analogue or digitised form and indexed using a text-based keyword or key-phrase index. In the short term such a facility would be a relatively easy thing to implement.

(d) Receiving and Transmitting Multimedia Documents

With more and more documents produced within computers and communicated over wide areas through communication lines, it is important that RMDBS be able to store, receive, and transmit such documents. So far, RMDBS is capable of transmitting documents to several local computers because the microcomputer on which it is implemented is on two local area networks, namely the IBM LAN and Ethernet. This is not enough. RMDBS would need to be extended to be able to transmit and receive documents worldwide. The most obvious solution to this problem is to use wide area networks, (like JANET, PSS, etc) in use today. Because

these networks have problems in handling multiple types of data, especially integrating voice and other types of data on a single transport system, using these types of networks might prove very difficult for multimedia documents.

A solution to the integration of voice and other types of data on a single transport system that RMDBS could take advantage of is The integrated services digital network(ISDN)[87]. This digital network is being defined for simultaneous carrying of digitised voice and a variety of data traffic on the same transmission link. Use of ISDN therefore has the following advantages: (i) the user does not have to buy multiple services to meet multiple needs, (ii) the communication makes virtually all data and information sources around the world easily and uniformly accessible.

(e) Animation

Another very interesting possibility that could be investigated with RMDBS is to see if some sort of animation can be done with the system. This will obviously need facilities to retrieve and change pictures very quickly. If sound for the animated pictures is also to be stored in the system, then tight coupling of the picture and sound retrieval will also be required.

Animation could be achieved in a variety of ways, for example, by storing image sequences, by computing images in real-time or by a combination of both of these methods. Each of these possibilities would need to be considered in order to extend RMDBS to cater for the storage of animated material.

9.3.2 User Friendliness

In addition to the extensions listed above to make RMDBS into a fully fledged multimedia database system, work in several other areas are important in order to assess and (if need be) improve its user-friendliness.

In order to achieve this, extensive field tests and trials need to be undertaken to study the user interface to see if it is truly user friendly. This will involve checking on the following:

(a) Interaction Method

So far, input to RMDBS is through the keyboard only. Would a mouse might make a better input device? This requires investigation.

The investigation can be done by implementing another version of RMDBS with the mouse as the input device. Users would then use both systems and compare them. From this comparison it would be possible to tell which system the users prefer, therefore, which type of input device is better according to the users.

(b) Using a Window Package for the Help Facility

Thus far, when the user requests for help, the help information occupies the whole screen. Experiments need to be done to see if users prefer to read the help information in a window occupying a portion of the screen only. This arrangement will leave the context the user was working on displayed on the portion of the screen not covered by the help information.

(c) Measuring the Usefulness of the On-line Help Facility.

Up to now it has been assumed that the help facility is fine. Tests need to be done to monitor the usefulness of the help facility to the user. This will answer questions such as; is the help information really being used? if so, how frequently? are there parts of the help facility that are used more than others? if so, which ones? If there are parts of the help facility that are used more than others, can the design be changed so that they are retrieved faster?

(d) Response Time Measurement

By means of a series of timing experiments it has been shown that the processing speed of RMDBS is adequate. But, is the response time of the system really adequate as far as the users are concerned? if not, how can it be improved?

To best way to investigate this is to let users evaluate every function in RMDBS during the field trials. Users would be asked to say whether they are satisfied with the response time

they get when they work with the function. If they are not satisfied, then more tuning is done to the function, for example, finding a better algorithm. This process could be done iteratively, that is, repeated until an optimal performance is obtained.

(e) Implementing a UIMS

Implementing a UIMS in RMDBS is desirable and necessary but it is better to defer such an undertaking for a few years. The main reason for postponing the implementation of a UIMS to RMDBS is that, the field of UIMS is still expanding and developing rapidly to an extent that no rules and guidelines are there to guide the practitioner. It is hoped that after waiting a few years the field will mature, that is, have clear and well defined guidelines on how to design UIMS. Implementing a UIMS to RMDBS then will, therefore, produce a better system than if the implementation is done now.

9.4 Final Conclusion

The design of a novel multimedia data base system has been described. The research involved in the creation of the system has extended quite considerably our experience and knowledge relating to the practical utility of relational DMBS as tools for the fabrication of sphisticated multimedia systems. The limitations of the work described in this thesis have been described as have its merits. A number of suggestions have been made for future work and extensions to RMDBS.

References

1. I. Palmer, *Database Systems: A Practical Reference,* C.A.C.I, London, 1975.

2. C. J. Date, *An Introduction to Data Base Systems,* Addison-Wesley, Reading, Mass., 1975.

3. J. P. Fry and E. H. Sibley, "Evolution of Data-Base Management Systems," *ACM Computing Surveys,* vol. 8 No 1, pp. 7-42, March 1976.

4. "The ANSI/SPARC DBMS Model," in *Proc. of 2nd SHARE Conf. Montreal, 1976,* ed. D. A. Jardine, North-Holland, New York, 1977.

5. *Data Base Management System Requirements,* GUIDE/SHARE Data Base Task Force, New York, November 1971.

6. "Introduction to 'Feature Analysis of Generalised Data Base Management Systems'," *Computer Bulletin,* vol. 4, CODASYL Systems Committee, April 1971. Also available in Comm. of ACM vol 14 no 5 (May 1971)

7. E. H. Sibley, "The Development of Data Base Technology," *ACM Computing Surveys,* vol. 8 No 1, pp. 1-5, March 1976. Guest Editor's Introduction

8. E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Comm. of ACM,* vol. 13 No 4, pp. 377-387, June 1970.

9. D. D. Chamberlin, "Relational Data-Base Management Systems," *ACM Computing Surveys,* vol. 8 No 1, pp. 43-66, March 1976.

10. J. Bradley, *File and Data Base Techniques,* Holt-Saunders, 1982.

11. E. F. Codd, "Normalised Data Base Structure: A Brief Tutorial," *Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control,* San Diego, November 1971.

12. E. F. Codd, "Further Normalisation of the Data Base Relational Model," *Courant Computer Science Symposia, Data Base System,* vol. 6, Prentice-Hall, New York, 1971.

13. B. Salzberg, "Third Normal Form Made Easy," *SIGMOD RECORD,* vol. 15 No 5, pp. 2-18, December 1986.

14. E. F. Codd, "A Data Base Sublanguage Founded on the Relational Calculus," *Proc. 1971 ACM-SIGFIDET Workshop on Data Description, Access and Control*, pp. 35-65, 1971.

15. E. F. Codd, "Relational Completeness of Data Base Sublanguages," in *Data Base Systems*, ed. R. Rustin, pp. 65-98, Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

16. R. W. Taylor and R. L. Frank, "CODASYL Data-Base Management Systems," *ACM Computing Surveys*, vol. 8 No 1, pp. 67-104, March 1976.

17. C. W. Bachman, "Data Structure Diagrams," *ACM SIGBDP*, vol. 1 No 2, pp. 4-10, 1969.

18. D. C. Tsichritizis and F. H. Lochovsky, "Hierarchical Data-Base Management Systems," *ACM Computing Surveys*, vol. 8 No 1, pp. 105-124, March 1976.

19. D. E. Knuth, *The Art of Computer Programming*, Addison-Wesley, Reading, Mass., 1968. Vol. 1

20. C. J. Date and E. F. Codd, *The Relational Model and Network Approaches: Comparison of the Application Programming Interfaces*, IBM Research.

21. A. S. Michaels, B. Mittman, and C. R. Carlson, "A Comparison of Relational and CODASYL Approaches to Data-Base Management," *ACM Computing Surveys*, vol. 8 No 1, pp. 125-151, March 1976.

22. G. D. Held, M. R. Stonebraker, and E. Wong, "INGRES- A Relational Data Base System," *Proc. AFIPS, AFIPS Press*, vol. 44, pp. 409-616, Arlington, Va., 1975.

23. W. Zook, K. Youssefi, P. Kreps, and G. Held, *INGRES Reference Manual*, Memo. ERL-M579, Univ. of Califonia, Berkeley, Califonia, 1977.

24. D. C. Tsichritizis and F. H. Lochovsky, *Data Base Management Systems*, p. 213, Academic Press, New York, 1977.

25. "A Survey of DBMS-Related Standardisation Activities," in *ISO/TC 97/SC 5/WG 5 Document N140*, ed. D. Brand and E. Peeters, April 1984.

26. *Framework Report on Database Management Systems*, AFIPS Press, Montvale, New Jersey, 1978. ANSI/X3/SPARC Study Group for Data Base Management Systems

27. T. W. Olle, "DBMS Standardisation - 1979 to 1983," *Computers and Standards*, vol. 2 No 2, pp. 119-126, 1983.

28. "Reference Model for DBMS Standardisation," *SIGMOD Record*, vol. 15 No 1, pp. 19-55, Data Base Architecture Framework Task Group(DAFTG), March 1986. of the ANSI/X3/SPARC Database System Study Group

29. E. Wong and W. B. Samson, "The Specification of Relational Database(PRECI) as an Abstract Data Type and its Realisation in HOPE," *The Computer Journal*, vol. 27 No 3, pp. 261-268, June 1986.

30. S. M. Deen, D. Nikodem, and A. Vashishta, "The Design of a Canonical Database System," *The Computer Journal*, vol. 24, pp. 200-209, 1981.

31. *Revelation User's Guide*, COSMOS Incorporated, 1985.

32. *Revelation Technical Reference*, COSMOS Incorporated, 1985.

33. R. Williams, G. M. Giddings, W. D. Little, W. G. Moorhead, and D. L. Weller, "Data Structures in Computer Graphics," in *Data Structures, Computer Graphics and Pattern Recognition*, ed. A. Klinger, K. S. Fu and T. L. Kunii, pp. 167-177, Academic Press, New York, 1977. Originally in Proc. Workshop on Data Bases for Interactive Design, University of Waterloo, Sept. 1975

34. J. D. Foley and A. van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley Publishing Company, Reading, March 1983. Referenced pages are 5, 359

35. W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill International Book Company, New York, 1975.

36. I. E. Sutherland, "SKETCHPAD: A Man-Machine Graphical Communication System," *Proc. of AFIPS 1963 Spring Conference*, pp. 329-346, 1963.

37. J. E. Scott, *Introduction to Interactive Computer Graphics*, John Wiley & Sons, New York, 1982.

38. M. Heck and M. Plaehn, "A Workstation Model for an Interactive Graphics System," *Comm. of ACM*, vol. 29 No 1, pp. 30-37, Jan 1986.

39. R. Williams, "A Survey of Data Structures for Computer Graphics Systems," in *Data Structures, Computer Graphics and Pattern Recognition*, ed. A. Klinger, K. S. Fu and T. L. Kunii, pp. 105-152, Academic Press, New York, 1977.

40. R. Williams, "On the Application of Relational Data Structures in Computer Graphics," in *Data Structures, Computer Graphics and Pattern Recognition*, ed. A. Klinger, K. S. Fu and T. L. Kunii, pp. 153-165, Academic Press, New York, 1977.

41. "CORE Graphics System," *Computer Graph*, vol. 13 No 3, pp. 91-96, Graphics Standards Planning Committee(SIGGRAPH-ACM), August 1979.

42. S. Harrington, *COMPUTER GRAPHICS A Programming Approach*, McGraw-Hill International Book Company, Singapore, 1985.

43. "Graphical Kernel System(GKS) - Functional Description," *ISO DP 7942*, International Standards Organisation, November 1982.

44. G. Penney, *Data Processing Case Histories*, NCC Publications, Sandbach, Cheshire, 1974. Learning to Ride the Computer

45. J. Martin, *Design of Man-Computer Dialogues*, Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

46. J. D. Foley and V. L. Wallace, "The Art of Natural Graphic Man-Machine Conversation," *Proc. IEEE*, vol. 62 No 4, pp. 462-470, April 1970.

47. E. F. Codd, "Recent Investigations in Relational Data Base Systems," *Proc. IFIP Congress 1974*, pp. 1017-1021, North-Holland, Amsterdam, 1974.

48. E. F. Codd, "Seven Steps to Rendezvous with the Casual User," in *Data Base Systems*, ed. J. W. Klimbie and K. L. Koffeman, pp. 179-199, North-Holland, Amsterdam, 1977.

49. N. Cercone and G. McCalla, "Accessing Knowledge Through Natural Language," in *Advances in Computers Volume 25*, ed. M. C. Yovits, pp. 1-99, Academic Press, Orlando.

1986.

50. M. M. Astrahan and D. D. Chamberlin, "Implementation of a Structured English Query Language," *Comm. of ACM*, vol. 18 No 10, pp. 580-588, October 1975.

51. L. Koved and B. Shneiderman, "Embedded Menus: Selecting Items in Context," *Comm. of ACM*, vol. 29 No 4, pp. 312-318, April 1986.

52. *Introduction to INGRES*, Relational Technology Inc., 1984.

53. D. H. H. Ingalls, "The Smalltalk Graphics Kernel," *Byte*, vol. 6 No 8, pp. 168-194, August 1981. Special issue on Smalltalk

54. N. McDonald and M. Stonebraker, *CUPID the Friendly Query Language*, Memo. ERL-M487, Electronics Research Lab., Univ. of Califonia, Berkeley, Califonia, October 1974.

55. M. M. Zloof, "Query-by-Example: The Invocation and Definition of Tables and Forms," *ACM Proc. International Conf. on Very Large Data Bases*, vol. 44, pp. 431-438, September 1975.

56. R. King and S. Melville, "Ski: A Semantics-Knowledgeable Interface," *Proc. of the Tenth International Conf. on Very Large Data Bases*, pp. 30-33, August 1984.

57. C. F. Herot, "Spatial Management of Data," *ACM Transactions on Database Systems*, vol. 5 No 4, pp. 493-514, December 1980.

58. W. Benn and B. Rading, "Retrieval of Relational Structures for Image Sequence Analysis," *Proc. of the Tenth International Conf. on Very Large Data Bases*, pp. 533-536, August 1984.

59. N. S. Chang and K. S. Fu, "Query-by-Pictorial-Example," *IEEE Transactions on Software Engineering*, vol. SE-6 No 6, pp. 519-524, November 1980.

60. T. L. Kunii, S. Weyl, and J. M. Tenenbaum, "A Relational Data Base Schema for Describing Complex Pictures with Colour and Texture," *Proc. of 2nd Joint Conf. on Pattern Recognition*, pp. 310-316, August 1974.

61. Y. E. Lien and D. F. Utter, Jr., "Design of an Image Database," *Proc. of Workshop on Picture Data Description and Management*, pp. 131-136, 1977.

62. D. Weller and R. Williams, "Graphic and Relational Data Base Support for Problem Solving," *ACM SIGGRAPH: Computer Graphics*, vol. 10, pp. 183-189, 1976.

63. N. S. Chang, *Image Analysis and Image Database Management*, Umi Research Press, Ann Arbor, Michigan, 1981.

64. A. GO, M. Stonebraker, and C. Williams, "An Approach to Implementing a Geo-data System," *Proc ACM SIGDA SIGMOD SIGGRAPH Workshop on Data Base for Interactive Design*, pp. 67-77, Sept 1975.

65. P. E. Mantey, J. L. Bennet, and E. D. Carlson, "Information for Problem Solving: The Development of an Interactive Geographic Information System," *IEEE International Conf. on Communication*, vol. II, pp. 44.3-44.6, Seattle, June 1973.

66. S. Gibbs, D. Tsichritzis, A. Fitas, D. Konstantas, and Y. Yeorgaroudakis, "Muse: A Multimedia Filing System," *IEEE Software*, pp. 4-14, March 1987.

67. M. Stefik, G. Foster, D. G. Bobrow, K. Kahn, S. Lanning, and L. Suchman, "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving," *Comm. of ACM*, vol. 30 No 1, pp. 32-47, January 1987.

68. D. Tsichritzis, S. Christodoulakis, P. Economopoulos, C. Faloutsos, A. Lee, D. Lee, J. Vandenbroek, and C. Woo, "A Multimedia Office Filing System," *Proc. of the 9th International Conf. on Very Large Data Bases*, pp. 2-7, 1983.

69. S. Christodoulakis, J. Vanderbroek, J. Li, T. Li, S. Wan, Y. Wang, M. Papa, and E. Bertino, "Development of a Multimedia Information System for an Office Environment," *Proc. of the tenth International Conf. on Very Large Data Bases*, pp. 261-271, Singapore, August 1984.

70. S. Christodoulakis, "Multimedia data Base Management: Applications and Problems: A Position Paper," *Proc. of ACM SIGMOD*, pp. 304-305, Austin Texas, May 28-32, May

1985.

71. M. Stonebraker, H. Stettner, N. Lynn, J. Kalash, and A. Guttman. "Document Processing in a Relational Database System," *ACM Transactions on Office Information Systems*, vol. 1 No 2, pp. 143-158, April 1983.

72. S. Christodoulakis, M. Theodoridou, F. Ho, M. Papa, and A. Pathria, "Multimedia Document Presentation, Information Extraction, and Document Formation in MINOS: A Model and a System," *ACM Transactions on Office Information Systems*, vol. 4 No 4, pp. 345-383, October 1986.

73. T. C. Waugh and J. McCalden, *GIMMS Reference Manual Release/Edition 4.5*, Edinburgh, 1983.

74. "1981 Census Data at NUMAC," *University of Durham Computer Centre Internal Memo*, Durham, 1985.

75. "SASPAC 3.0," *University of Durham Computer Centre Internal Memo*, Durham, May 1983.

76. *The Smart Spreadsheet with Graphics*, Innovative Software, Inc., July 1985.

77. *TURBO PASCAL Version 3.0 Reference Manual*, Borland International Inc., Scotts Valley, 1985.

78. K. Jensen and N. Wirth, *Pascal User Manual and Report*, New York, 1978.

79. *TURBO Graphix Toolbox*, Borland International Inc., Scotts Valley, July 1985.

80. *The Smart System Manual*, Innovative Software, Inc., 1986.

81. *Microsoft Windows User's Guide*, Microsoft Corporation, 1985.

82. B. Shneiderman, *Designing the Interface - Strategies for Effective Human-Computer Interaction*, Addison-Wesley, Reading, Mass., 1982.

83. R. Rubinstein and H. Hersh, *The Human Factor - Designing Computer Systems for People*, Digital Press, Burlington, Mass., 1984.

84. J. Coutaz, "Abstractions for User Interface Design," *IEEE Computer*, vol. 18 No 9, pp. 21-34, 1985.

85. D. R. Olsen, G. Buxton, R. Ehrich, D. Kasik, J. Rhyne, and J. A. Sibert, "A Context for User Interface Management," *IEEE Computer Graphics and Applications*, vol. 4 No 12, pp. 33-42, 1984.

86. C. Faloutsos, "Access Methods for Text," *ACM Computing Surveys*, pp. 49-74, March 1985.

87. W. Stallings, *Data and Computer Communications*, Macmillan Publishing Company, New York, 1985.

Appendix 0

Hardware Configuration

The Relational Multimedia Data Base System is implemented on the following hardware configuration:

1. An IBM PC-XT/286 running DOS 3.20 with one floppy disk and a 20Mb hard disk, with two serial ports and a parallel port.
2. Attached to one of the serial ports is a Microsoft serial mouse
3. A Hercules Monochrome Graphics Card supports both text and graphics display.
   When in Graphics mode, the card offers two (712 X 350 pixels) graphics pages. One of these is displayable and the other in RAM.
4. The XT is on two local networks: the IBM LAN and on Ethernet via a SUN computer

Another equipment used is the ICL PERQ digitiser. This digitiser*, consists of:

a digitising table

a digitiser 'black box' attached to the underside of the table

The ICL PERQ comprising of processor, a hard disc, a display, and a floppy disk drive

The digitising software on the PERQ produces a file of coordinates (accurate to 0.01mm) on the hard disk. This is transformed to a floppy disc for use as required.

---

* K. G. Middleton *DIGITISING with the ICL PERQ*, UDCCC Guide No 5, University of Durham Computer Centre, Internal Memo.

Coordinates are produced in form:

(X, Y, FLAG)

The coordinates are sent to the processor from the table by moving either a cursor or a stylus over the table.

By digitising with different buttons on the cursor pad, a varying FLAG may be stored with each digitised point. Commonly, flags are used to signal either a pen colour, a change from 'join' to 'move' in the plotting sequence, a request to enter a label to be plotted at the point, etc.

Appendix 1

The Revelation DBMS

The Revelation DBMS has five major components: R/LIST, R/DESIGN, R/BASIC, R/LAN, and the editors, R/EDIT and R/TEXT. This appendix discusses the main features of R/LIST, R/DESIGN and R/BASIC as used in this thesis.

R/LIST

R/LIST is the name of Revelation's query (inquiry) language. R/LIST allows the user to retrieve data from a database using 'plain English' sentences to query the database.

R/LIST is a dictionary driven language. That is, the words that are used to compose the commands in the language are contained in one of the two dictionaries.

The master dictionary, called the VOCabulary file, contains descriptions of the common verbs (for example LIST, SORT), the connectives(OR and AND) and the control modifiers(for example BY, WITH, WHICH). These are words that relate to all the files.

The data dictionary relates only to the specific file; it contains descriptions of the data fields that are in that specific file. These descriptions specify fields, display formats, functional calculations, interfile retrieval operations and other technical information for each field in the file.

R/LIST Verbs

The user can invoke any valid R/LIST verb by typing and entering the word at the TCL colon prompt. All verbs are similar in format. These are the verbs found in R/LIST.

LISTFILES

Allows the user to see the list of accessible files

LIST

This verb lists the contents of any file.

SORT

Performs exactly like LIST except that the records in a file are output in the order specified by the user.

SELECT

Also performs like the LIST verb except that the records are selected instead of just listed.

SSELECT

Selects and sorts the records.

COUNT

COUNT simply counts the number of records that meet the conditions set up by the user.

SUM

The SUM verb is used when a field is to be totaled.

General form of R/ LIST Commands

The following is the general form of R/LIST commands:

verb {DICT} filename {record.list}{selection.criteria} {output.specs}{options}

verb - is the operation required

{DICT} - specifies that the dictionary of the file rather than the file is to be processed. This parameter is optional

filename - one and only one may be given in a command

{record.list} - Optional fields for eligible records.

{selection.criteria} - Optional limit for eligible records

{output.specs} - Optional fields for output for example BREAK-ON

{options} - Optional command option for example PRINTER or SCREEN.

The following is a list of the compare words that instruct the system to compare one value against another:

```
=      EQ  EQUAL {TO}, THAT IS        LIKE    ARE
<      LT  LESS THAN                  BEFORE  UNDER
< = =< LE  LESS THAN {OR} EQUAL TO
# <>   NE  NOT, NOT EQUAL {TO}, DOESNT  ARENT   INST
> = => GE  GREATER THAN {OR} EQUAL TO   FROM
>      GT  GREATER THAN, LATER THAN     AFTER   OVER
          STARTING {IN}, ENDING {IN}    CONTAINING
          MATCH, MATCHES
```

The General Format of the compare clause is:

WITH < dict.field> < compare.word> < value>

For example

WITH AMOUNT GREATER THAN 9000

Any number of compare clauses may be used in an R/LIST command to refine the selection criteria, but must be connected with a connective AND or OR.

Examples of R/LIST commands are:

```
LIST CUSTOMERS WITH INVOICE.AMOUNT < 500
COUNT CUSTOMERS
SORT CUSTOMERS COMPANY STATUS STATUS.DESC (P)
```

R/DESIGN

R/DESIGN is Revelation's 4th generation applications language. Below are specific

details necessary to build and use R/DESIGN generated custom data entry screens and reports. The topics that the user must follow when creating a database are:

DEF File Definition and Creation Program

BUD Dictionary Builder

PGMR Program Designer

SEL Field Selection Screen

SCR.GEN Default Screen Generator

SCR Customising Screens

BLD.MENU Build a Menu

The main routine in R/DESIGN is BUD, which is used to build a data dictionary. Up to 15 items can be defined for each field in the file. The following is a sample BUD screen.

```
BUD        BUILD DICTIONARY RECORDS

01 File Name
02 Field Name
03 Single/Multivalued
04 Field Type (F,S,G)
05 Field Number              06 Which Part of Key
07 Output Conversion
   (MONEY,DATE,TIME)

08 R/BASIC Formula
09 Justification             10 LIST Display Length
11 Edit Patterns
12 Bottom Screen Prompt
13 LIST Column Heading
14 Source Information
15 Description
```

The BUD Screen

Though items 14 and 15 on the BUD screen are not used by R/LIST, when given they help in (automatic) documentation.

R/BASIC

R/BASIC, Revelation's programming language, is basically BASIC but with extended features for:

(i) file handling

(ii) record locking (for the LAN version)

(iii) executing (R/LIST) statements by sending a command to the TCL processor. This is done by the PERFORM and EXECUTE statements.

(iv) executing DOS commands through the PCPERFORM statement.

Appendix 2

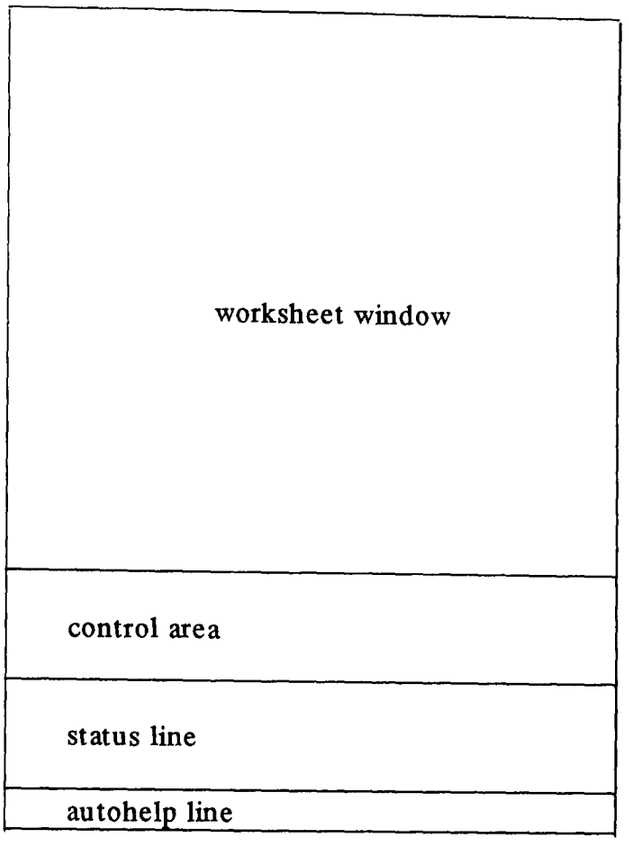
Smart Spreadsheet

The Smart Spreadsheet program is part of the integrated Smart System Software consisting of the Communications-Manager, Data-Manager. Time-Manager. and the Wordprocessor. Because the Smart System is modular in nature, each application program has its own unique structure. This permits the user to run the application program independently or to use it together with any combination of other Smart Applications.

Project processing allows the user to automate sequences of commands and functions by placing a program in remember mode. The program records all commands and functions the user performs, and thereafter, will repeat the sequence automatically when that project is executed.

The Spreadsheet

The screen display for the spreadsheet can be divided into four areas



The Spreadsheet Screeen

Autohelp Line

Is at the bottom of the screen, shows a single line of help information related to the highlighted command

The Status Line

Is just above the autohelp line. It contains information relating to the status of the worksheet and computer. It shows name of worksheet, current location in worksheet, font, etc.

The Control Area

Here the user selects commands, responds to prompts and enters information.

The Worksheet Window

A worksheet is a large grid with 9999 rows and 999 columns; the intersection of a row and a column is called a cell. Each cell can be thought of as a particular row/column combination

(for example r1c1). Numbers, text, and formulas can be entered into any cell. In addition to referring to cells on a worksheet, a block of cells can be referenced.

The GRAPHICS Command - an overview

GRAPHICS allows the user to define and display graphs that reference worksheet data. This command also includes an "edit screen" feature that permits the drawing of lines and characters on an existing graph or blank screen.

The available graph types are bar-line graphs, pie charts, hi-low graphs, layer graphs, histograms, and X-Y graphs. Each graph type offers various options to enhance the basic graph.

To create a graphics representation of worksheet data, the GRAPHICS DEFINE command is used to produce a graph definition file(gdf). In the process of producing a gdf, information on as many as three screens must be entered:

1. the General Graph Definition screen
2. the General Definition Screen for the graph type being created
   (bar/line, pie chart, etc.)
3. the Options Definition Screen for the graph type being created

The formats for the GRAPHICS commands are:

GRAPHICS DEFINE (gdf)

GRAPHICS GENERATE (gdf) COLOR SCREEN (scn-optional)

GRAPHICS GENERATE (gdf) BLACK/WHITE (scn-optional)

GRAPHICS PLOT (gdf) FULL-PAGE

GRAPHICS PLOT (gdf) QUADRANT (quadrant #)

GRAPHICS MATRIX-PRINT

GRAPHICS UNDEFINE

GRAPHICS SLIDESHOW

GRAPHICS EDIT (scn-optional)

GRAPHICS VIEW (scn) CURTAIN (type)

GRAPHICS VIEW (scn) FADE-IN

GRAPHICS VIEW (scn) INSTANT

Where:

scn is a graph screen file

type is a curtain type 1-3

Project Processing

Project processing refers to the building and execution of project files. A project file is a collection of commands which perform some predetermined activity. These commands are executed in sequence when the project is executed.

With the REMEMBER command the user can instruct the Smart System to remember a procedure as it is performed. The project file is automatically created after the user has gone through a series of tasks the first time. When the user wishs to perform the same tasks at a later date, he/she simply executes the project file.

A project file is much more than a repetition of a fixed series of predefined tasks. The Smart Programming Language includes commands to generate menus, make decisions. manipulate variables and more. Since project files are created using the Smart language rather than by saving key-strokes, they are easy to review and edit.

Building Projects

The key to creating a project is the REMEMBER command. When it is selected, the following option list is displayed:

select option: Compile Delete Edit Finish Help Print Start

The DELETE option is used to erase a project file from the disk.

REMEMBER START activates and REMEMBER FINISH deactivates the remember mode. The REMEMBER EDIT option accesses the project editor.

A project file may also be created with any text editor or word-processor that can read and write standard ASCII text files. If the user creates a project in this manner, the REMEMBER COMPILE command is used to create the run-time version of the project file.

Accessing the Spreadsheet

The are two standard ways of accessing the spreadsheet from the terminal; first through the Smart system or directly into the spreadsheet.

In the first case, the user types SMART at the terminal, after which the Main Smart Menu will appear. Select SPREADSHEET on command list 1.

In the second case, the user types SMART S. This bypasses the Main Smart Menu and gets him/her straight into the Spreadsheet.

In this thesis the command used is:

SMART S -Pproject.file

which gets the user straight into the Spreadsheet, and executes the project specified to automatically:

load the required worksheet

start graphics definition

generate the graph after its definition

prompt the user for file name to save graph

quit Smart when graph has been saved

Appendix 3

Microsoft Windows

Microsoft Windows is an extension of the DOS operating system. It comes equipped with built-in applications: CLIPBOARD, NOTEPAD, CARDFILE, CALENDAR, CLOCK, TERMINAL, and PAINT. Each is like a standard desk-top aid. Windows allows the user to integrate these tasks, optionally running several programs at once.

You can switch between programs with a couple of key-strokes or a click of a mouse. And since the user never have to quit a program, s/he can continue from where s/he left off.

The Mouse

The Microsoft Mouse is a pointing device used to move images on the display, edit or format text, to create or edit pictures, and to choose commands in certain mouse supported applications. The computer responds to movements made by the mouse, which with a tracking resolution of 200 dots/inch, fairly good images can be drawn.

The PAINT Program

Windows Paint is a drawing tool that is a Windows applications, that is, it was designed especially to run with Microsoft Windows.

When Windows Paint is started, it gives a paint canvas and a palette of 24 drawing tools, plus the menu:

File Edit Font Fontsize Style Palette Options

Each option of the above menu has a set of options under it. They appear when the mouse is clicked on the option. Below is the list of such options:

FILE	EDIT	FONT	FONTSIZE	STYLE	PALETTE	OPTIONS
new	undo	system	12	plain	patterns	zoom in
open	erase	terminal	24	bold	line widths	zoom out
save	cut		36	italic	brush shapes	no grid
save as	copy		48	underline	tools	fine grid
print	clear		60	outline		medium grid
	invert		72	strike out		coarse grid
	trace edges		84	align left		edit pattern
	flip vertical			align center		for printer
	flip horizontal			opague		for screen
				transparent		

To draw a picture the user chooses drawing tools from the palette, for example pencil(for free hand sketch), line(for drawing lines), circle(for drawing circles), text(for writing text) etc.

After drawing the picture, it can be saved in a file by clicking the mouse on FILE and choosing the required option. After which the program will prompt for the file name of the file to save the picture.

Appendix 4

Turbo Pascal

Turbo Pascal

The Turbo Pascal system is implemented for the PC-DOS, MS-DOS, CP/M-86 and CP/M-80 operating systems.

Turbo Pascal closely follows the definition of Standard Pascal as defined by K. Jensen and N. Wirth in the "Pascal User Manual and Report". In addition to the standard, a number of extensions are provided, such as:

absolute address variables

bit/byte manipulation

direct access to CPU memory and data ports

dynamic strings

free ordering of sections within declaration part

full support of operating system facilities

in-line machine code generation

include files

logical operations on integers

overlay system

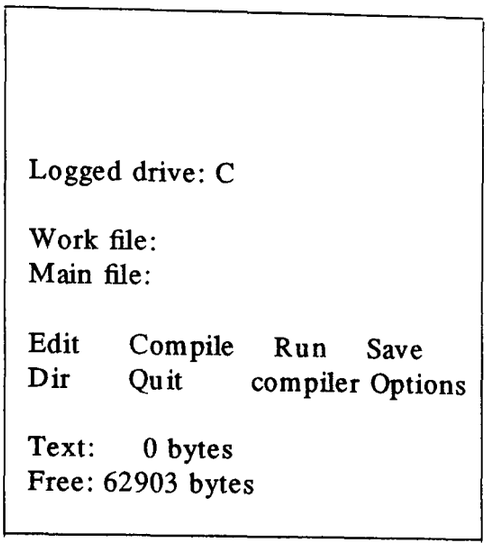
program chaining with common variables

random access data files

structured constants

type conversion functions

Turbo Pascal is started by typing TURBO at the terminal. After this it comes with the menu shown below.



Main Turbo Pascal Menu

The above values for Logged drive and memory use are for illustration purposes only; the corresponding values shown will be the actual values for the computer in use.

Each command in the menu is executed by entering the associated capital letter (if the terminal has highlighting capabilities, (this letter is highlighted).

The Turbo Pascal editor(similar to Word Star) appends the extension ".PAS" to all source file names. For the corresponding binary programs the extension is ".COM" if they are not chained programs. If they are chained then the extension is ".CHN". Chained programs can be executed in other programs with common data.

To write Turbo Pascal programs with graphics, the Turbo Graphix Toolbox, discussed below, is used.

Turbo Graphix Toolbox

The Turbo Graphix Toolbox contains a set of procedures and functions to be used in Turbo Pascal programs. With the aid of this Graphix Toolbox, high-resolution monochrome graphics can be developed for the IBM-PC and PC-compatible computers(using either an IBM or Hercules card), and the Zenith-100 computer.

The Turbo Graphix Toolbox is a versatile package, designed for both simple and

complicated graphics applications. Simple procedures allow the user to draw:

points

lines

rectangles

ellipses

circles

High-level procedures allow the user to create more complex graphics that are often needed in business and scientific applications:

labelled pie charts

bar charts

a variety of curves

curve fitting

line and solid modelling

polygons

Screen Management

All drawings can be displayed either on the full screen or in windows. Drawing can also be done in RAM(virtual) screen in memory without display, and move the resulting image to the displayed screen. The Graphix Toolbox routines allow the user to save and load either of these screens to and from disk, and restore them when needed. The screen contents of one can also be swapped with the contents of the other. The screen currently being drawn is called the active screen.

Windows Management

A window is any area of the screen that the user defines as the drawing area. Several windows, containing different drawings, can be defined and then displayed simultaneously on the screen. Each window can be moved independently of other windows, placed on top of others, and stored to, recalled from, or erased from memory. The user can draw borders,

incorporate high-quality text, and label windows with headers or footers. The window currently being drawn is called the active window.

A window can be specified to be almost any size, from the whole screen to 1 vertical pixel by 8 horizontal pixels. The window area is defined by specifying the X and Y coordinates of its upper left and lower right corners, with Y values measured in 1-pixel units and X in 8-pixel units. These coordinates are called window definition coordinates.

Further to specifying the window definition coordinates when defining a window, a window is associated with a number, the window number. The window number is unique for each window and it is used to identify a window.

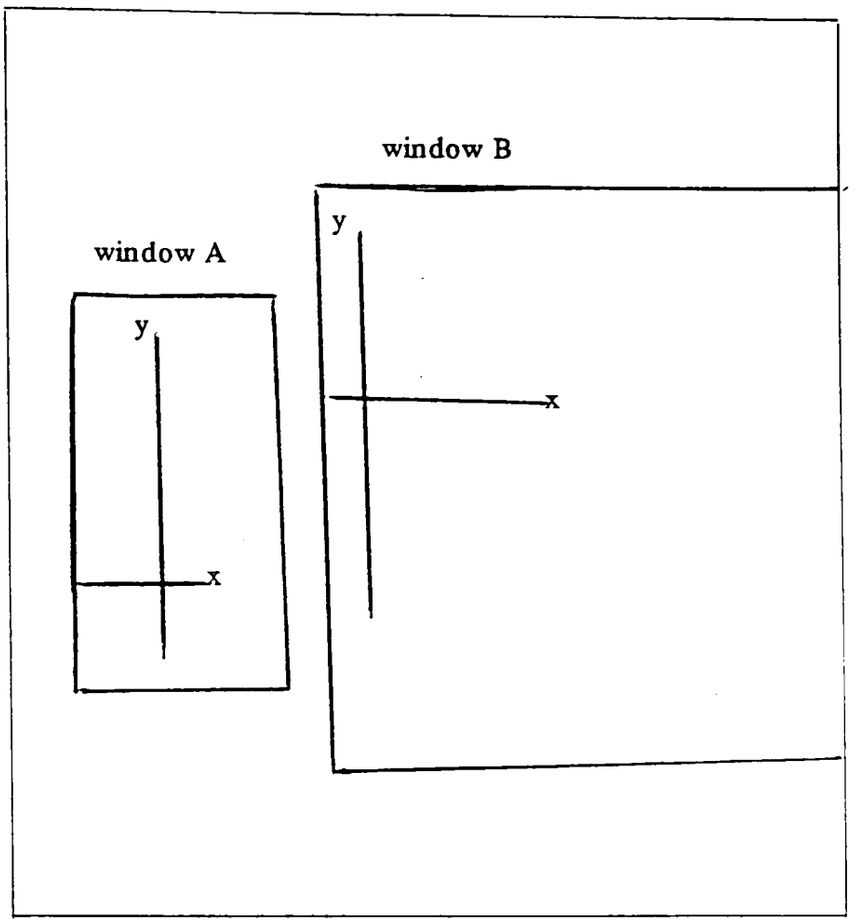
In the Turbo Graphix Toolbox the following are distinguished: screen coordinates, world coordinates, and windows as explained below.

The absolute screen coordinate system refers to the entire monitor screen; coordinates (0,0) are in the upper left corner of the screen, with the X coordinates increasing to the right up to 719, and the Y coordinates increasing downward up to 349.

A world coordinate system is an arbitrary system that is specified to accommodate any particular application. World coordinates are completely unrelated to the screen coordinates.

A window is any area of the screen that is defined as a drawing area. Each window can be moved independently of other windows, stored, recalled, or erased from memory. World coordinates can be scaled so that they fit correctly into any defined window.

The figure below illustrates two windows A and B, each with its own world coordinates.



Windows & World Coordinates

To use the Turbo Graphix Toolbox, system files are incorporated in the Turbo Pascal program through the "include directive".

Every Turbo Graphix program must include the following system files, in the order given below.

{$I TYPEDEF.SYS}

{$I GRAPHIX.SYS}

{$I KERNEL.SYS}

Other include files together with their uses are:

{$I WINDOWS.SYS} - for applications needing windows

{$I CIRSSEGM.HGH} - circles and ellipses

{$I PIE.HGH} - for labelled pie charts

{$I HATCH.HGH} - when hatching histograms

{$I HISTOGRM.HGH} - when histograms, bar graphs, etc are needed

{$I POLYGON.HGH} - if using polygons

{$I AXIS.HGH} - if axes are to be drawn/labelled

{$I SPLINE.HGH} - when curve fitting

{$I BEZIER.HGH} - needed for line and solid modeling

{$I MODPOLY.HGH} - for polygons

These files are included only if the functions they offer are needed in the program.

Images drawn by the Graphix Toolbox can be saved by either saving window or screen contents.

The SAVESCREEN procedure is used to store the active screen as a file on a disk. The single parameter passed in the routine specifies the file name in which to save the screen contents.

The contents of a given window can be saved in a file on the disk by the SAVEWINDOW procedure. This procedure takes two parameters; the window number of the window to be saved and name of file in which to save the window contents.

Appendix 5

The DEC LNO3 Laser Printer

The LNO3 Electronic Printer is a table-top, non-impact page printer that uses laser imaging and xerographic printing techniques. The printer prints letter quality images on cut sheet blank or preprinted paper, in two sizes, at a rate of 8 pages per minute.

The printer is capable of printing in two different orientations: portrait and landscape. For each printing orientation, there are several mono-spaced fonts permanently stored in ROM, 32 fonts in all: 24 portrait and 8 landscape. Further to the 32 fonts in ROM, up to 32 other fonts can be loaded from the host computer and stored in RAM.

Data Interface

The LNO3 data interface is a serial interface, that conforms mechanically to RS-232-C, functionally to a data subset of RS449, and electrically to RS423. The interface connection to the LNO3 is via a 25-pin, male receptacle mounted on the rear of the cabinet.

Character Code Processing

The printer processes characters in accordance with the ANSI standard X3.4-1977, and the ISO DIS 2022-1984. The coded characters are processed based on category:
printable or graphic characters, control characters, escape sequences, control sequences, and control strings.

Printable ASCII characters are printed as they are received by the printer. The escape sequences, control sequences, and control string consist of multiple characters/bytes grouped together to indicate a specific control function. See figure 4.1, 4.2, tables 4.1, 4.2 and tables B-1/14 of its Programmer Reference Manual* for a complete list of the character groupings.

---

* LNO3 Programmer Reference Manual, *Educational services DEC, Massachusetts. January 1985.*

Sixel Graphics

The LNO3 produces graphic images by the protocol known as the sixel protocol. The sixel protocol allows devices to receive and print black and white bit-map data at various sizes over a stream oriented communications line. Six bits of each 7 or 8 bit character code are used to represent bit-map data, allowing the remaining values to be used to control the context of the communications line and to fit within the ANSI text syntax.

The smallest displayable unit in the sixel protocol is a dot. This can be a light dot on a screen, an ink dot on the paper. Dots can be round, oval, square, rectangular, small or big.

A sixel is a group of 6 vertical pixels represented by 6 bits within a character code. A one value for a bit indicates that a pixel-spot will be placed at the corresponding grid position; a zero value is not written. Upon entering sixel mode, the current sixel position is determined from the ANSI text position and is called the graphic left margin. As each sixel is printed, the sixel active position is advanced to the next horizontal grid position. Positioning is always relative to the sixel active position, and cannot go backwards except for returning to the graphic left margin via the graphic carriage return and the next line commands.

The graphics next line command causes the sixel active position to move to the left margin and down one row of sixels(six actual grid units).

Appendix 6

RMDBS Main Routine

```
****
**** THIS IS THE MAIN-LINE PROGRAM TO CONTROL
**** ALL FUNCTIONS IN RMDBS
****
****
PERFORM "WELCOME"
PERFORM "CONTINUE"
PERFORM "INSTRUCTIONS"
PERFORM "CONTINUE"
START:
PRINT @(-1):@(-2)
PRINT ""
PRINT ""
PRINT "NO.  FUNCTION"
PRINT " 0  Get HELP Information"
PRINT " 1  VIEW: display a picture on the screen"
PRINT " 2  PUT.SCRPIC: insert a screen picture into database"
PRINT " 3  PUT.LSRPIC: insert a laser printable picture into database"
PRINT " 4  PUT.PROG: insert a binary program into database"
PRINT " 5  HARD.COPY: print a laser printable picture "
PRINT " 6  PRINT: a screen picture onto laser printer"
PRINT " 7  SCALE: scale a picture up/down by a constant factor"
PRINT " 8  OVERLAY: merge two pictures to produce a third in a DOS file"
PRINT " 9  SMART: run Smart Spreadsheet to draw/edit picture"
PRINT "10  WIN: execute WINDOWS PAINT to draw/edit a picture"
PRINT "11  DRAWON: retrieve data and draw it on a picture"
PRINT "12  MAIN.MENU: perform the main Revelation MENU"
PRINT "13  DEL.PICS: delete 'least used' pictures to give room on disk"
PRINT "14  CHANGE.PARS: change system parameters"
PRINT "15  SEE: what programs, screen or laser pictures are in database"
PRINT "16  DELETE: a program, screen or laser picture from the database"
PRINT "17  EXIT: to TCL"
PRINT
CHOOSE:
PRINT "PLEASE TYPE CHOICE NO."
INPUT CHOICE
IF CHOICE <  0 OR CHOICE >  17 THEN
PRINT "WRONG CHOICE"
PRINT "CHOICE MUST BE BETWEEN 0 AND 17"
GOTO CHOOSE
END
BEGIN CASE
CASE CHOICE =  0; GOSUB HELPER
CASE CHOICE =  1; GOSUB DISPLAYPIC
CASE CHOICE =  2; GOSUB INSERTPIC
CASE CHOICE =  3; GOSUB INSERTLSR
CASE CHOICE =  4; GOSUB INSERTPRG
```

```
CASE CHOICE =  5: GOSUB LASER.ONE
CASE CHOICE =  6; GOSUB PRINT.SCR
CASE CHOICE =  7; GOSUB SCALER
CASE CHOICE =  8; GOSUB OVERLAYER
CASE CHOICE =  9; GOSUB SMARTS
CASE CHOICE =  10; GOSUB WINPAINT
CASE CHOICE =  11; GOSUB DRAWON
CASE CHOICE =  12; GOSUB REV
CASE CHOICE =  13; GOSUB DELPICS
CASE CHOICE =  14; GOSUB CHANGEPARS
CASE CHOICE =  15; GOSUB SEE
CASE CHOICE =  16; GOSUB DELT
CASE CHOICE =  17; GOTO LEAVE
END CASE
GOTO START
****
HELPER:
PERFORM "INSTRUCTIONS"
PERFORM "CONTINUE"
RETURN
**** display screen picture
DISPLAYPIC:
PRINT " ENTER NAME OF PICTURE TO BE DISPLAYED"
INPUT PICNAME
PERFORM "VIEW ":PICNAME
RETURN
**** insert in screen picture
INSERTPIC:
PRINT " ENTER PICTURE NAME"
INPUT PICNAME
PRINT " ENTER FULL PATH NAME OF PICTURE FILE"
INPUT PICFILE
PERFORM "PUT.SCRPIC ": PICNAME :" ": PICFILE
RETURN
**** insert a laser picture
INSERTLSR:
PRINT " ENTER PICTURE NAME"
INPUT PICNAME
PRINT " ENTER FULL PATH NAME OF PICTURE FILE"
INPUT PICFILE
PERFORM "PUT.LSRPIC ": PICNAME :" ": PICFILE
RETURN
**** insert a binary program
INSERTPRG:
PRINT " ENTER PROGRAM NAME"
INPUT PICNAME
PRINT " ENTER FULL PATH NAME OF PROGRAM FILE"
INPUT PICFILE
PERFORM "PUT.PROG ": PICNAME :" ": PICFILE
RETURN
**** print on laser a laser picture
LASER.ONE:
PRINT " ENTER NAME OF PICTURE TO BE PRINTED"
INPUT PICNAME
```

```
PERFORM "HARD.COPY ":PICNAME
RETURN
**** print on laser a screen picture
PRINT.SCR:
PRINT "ENTER NAME OF PICTURE TO BE PRINTED"
INPUT PICNAME
PERFORM "PRINT ":PICNAME
RETURN
**** scale a screen picture
SCALER:
PRINT " ENTER NAME OF PICTURE TO BE SCALED"
INPUT PICNAME
PERFORM "SCALE ":PICNAME
RETURN
**** overlay two pictures to produce a third
OVERLAYER:
PRINT " ENTER NAME OF FIRST PICTURE"
INPUT PIC1
PRINT " ENTER NAME OF SECOND PICTURE"
INPUT PIC2
PRINT " ENTER NAME OF NEW PICTURE"
INPUT NEWPIC
PERFORM "OVERLAY ":PIC1:" ":PIC2:" ":NEWPIC
RETURN
**** execute smart
SMARTS:
PERFORM "EXE.SMART"
RETURN
**** execute windows
WINPAINT:
PERFORM "EXE.WIN"
RETURN
**** draw-on a picture
DRAWON:
PRINT " ENTER NAME OF PICTURE TO DRAW ON"
INPUT PICNAME
PERFORM "DRAWON ":PICNAME
RETURN
**** delete "least used" picture
DELPICS:
PERFORM "DEL.PICS"
RETURN
**** change system parameters
CHANGEPARS:
PERFORM "CHANGE.PARS"
RETURN
**** see names of programs, pictures in db
SEE:
PERFORM "LIST.NAMES"
RETURN
**** delete unwanted program or picture
DELT:
PERFORM "DEL.ONE"
RETURN
```

```
**** descend to the Main Revelation Menu
REV:
PERFORM "MENU"
RETURN
**** leave rmdbs
LEAVE:
STOP
****
END
```