THE DEVELOPMENT OF AN INTEGRATED DATABASE FOR MANUSCRIPTS BASED ON THE REQUIREMENTS OF THE ANGLO-AMERICAN CATALOGUING RULES, SECOND EDITION (AACR2R) AND THE NATIONAL REGISTER OF MANUSCRIPTS (NAREM)

by

KARUNANANDA CHETTY

Submitted in part fulfilment of the requirements for the degree of Master of Information Studies in the Department of Information Studies at the University of Natal, Pietermaritzburg.

Date Submitted: 31 December 1993
ABSTRACT

The study examines the requirements of a catalogue entry in terms of the archival tradition on the one hand and the requirements of the library and information services on the other. Each tradition has its own set of principles and practices which determine the way manuscript collections are arranged and described. In the archival tradition, different kinds of finding aids, for example, inventories, guides, lists and catalogues, are created and maintained to serve as finding tools, while in the library profession, the catalogue remains the chief finding tool.

The requirements of NAREM (National Register of Manuscripts) in South Africa is in keeping with the archival tradition while the requirements of the AACR2R (Anglo-American Cataloguing Rules, second edition) is in keeping with standards adopted by library and information services. In the case of institutions wanting to report to NAREM while at the same time wanting to retain their current library practices, a composite model reflecting both the requirements is desirable so that common elements are identified and duplicate elements, eliminated, where possible.
Catalogue entries for each type of requirement can be produced from such a composite model. Hence the study developed and tested a composite model to meet both requirements.

In order to arrive at a composite model, the technique of database design was employed whereby the requirements were first ascertained and suitable entity-relationship models constructed. The Entity-Relationship diagrams for each type of requirement were combined into a composite model (Conceptual) which were then mapped onto a Dbase III+ relational model. Thereafter, the model underwent further refinement and fine-tuning to arrive at an efficient and economical solution to the problem at hand.

The resultant Dbase III+ model was then tested using suitable Dbase and Clipper (compiler for Dbase III+) programs, samples for which were taken from five different sources. Arising from the test data, the design techniques used, the Dbase characteristics and the composite model were assessed. Thereafter various conclusions based on the study were made. The recommendations that follow point the way to future areas of research.
ACKNOWLEDGEMENTS

The researcher would like to thank many people, who in one way or other assisted in completion of this project. Special thanks go to the following persons:

Professor W J Horton and Dr A Kaniki for their assistance and encouragement during the process of planning and writing of this thesis.

Special thanks goes to the staff of the Computer Centre of the University of Durban-Westville, for their assistance and willingness to provide guidance and an efficient service at all times.

To Kassarlall Harryparshad of the inter-library loans section of the University of Durban-Westville Library for being so ever willing to request items from other universities and institutions.

Special thanks goes to Vimla Frank and Kishore Anirudh for proofreading of the thesis.

Finally, to my family for the sacrifice they made during my long hours of study.

iii
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE SHEET</td>
<td>i</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>iii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>iv</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td></td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS</td>
<td>x</td>
</tr>
<tr>
<td>LIST OF ILLUSTRATIONS</td>
<td>xi</td>
</tr>
</tbody>
</table>

## CHAPTER 1 - INTRODUCTION

1.1 Introduction                              | 1    |
1.2 Computers, Databases and Cataloguing Codes| 4    |
1.3 Aims and objectives of the Study          | 7    |
1.4 Literature Review                         | 7    |
1.5 Method of Developing and Testing the      | 14   |
    Composite Model                            |
1.6 Scope and Limitations of Study           | 22   |
1.7 Reference Technique                       | 23   |

## CHAPTER 2 - DATABASE MANAGEMENT SYSTEMS (DBMS)

2.1 Introduction                              | 26   |
2.2 Why use a DBMS                            | 29   |
2.3 Components of a DBMS                      | 39   |
2.4 DBMS Models                               | 45   |
2.4.1 Relational Model                        | 48   |
2.4.1.1 Structure                            | 48   |
2.4.1.2 Data Integrity 52
2.4.1.3 Manipulation 53
2.5 Dbase DBMS 57
2.5.1 Primary Interfaces 58
2.5.2 Dictionary 60
2.5.3 Index Definition 63
2.5.4 Views and Relationships 67
2.5.5 Query and Data Manipulation 73
2.5.6 Screen and Report Design 77
2.5.7 Dbase Programming Language 80
2.5.8 Drawbacks of Dbase III+ 90

CHAPTER 3 - DATABASE DESIGN PROCESS

3.1 Introduction 98
3.2 Phases in Database Design 98
3.3 Requirements Definition 100
3.4 Conceptual Design 101
3.4.1 Data Normalisation 103
3.4.2 Entity-Relationship Model (E-R Model) 117
3.4.2.1 Entities 118
3.4.2.2 Relationships 120
3.4.2.3 Value and Limitations : E-R Model 126
3.5 Implementation Design 127
3.5.1 Steps in implementing the Relational Model 128
3.5.2 Specifying Requirements and Database Integrity 131
3.6 Physical Design 133
3.6.1 File Organisation: Access and Processing 134
3.6.2 Records 140
3.6.2.1 Record Structures 141
3.6.2.1.1 Fixed Length Record Format 142
3.6.2.1.2 Variable Length Record Formats 144

CHAPTER 4 - MANUSCRIPT CATALOGUING 155
4.1 Introduction 155
4.2 Objectives of the Catalogue 156
4.3 Need for a Cataloguing Code 157
4.4 The AACR2R 159
4.4.1 Composition and construction of a Catalogue Entry 160
4.4.2 Cataloguing Principles: Item and Collection Levels 163
4.4.3 Analytics 164
4.4.4 Alternate and Main Entry Approaches 167
4.5 Main Entry and Main Entry Heading 170
4.6 ISBD 173
4.6.1 Nature of the ISBD 175
4.6.2 Levels of Description 178
4.7 Manu Script Cataloguing According to AACR2R 180

vi
4.8 Extended Model for Bibliographic Elements 184
4.9 Entity Analysis and Normalisation of AACR2R Elements 190
  4.9.1 Title and Statement of Responsibility Area 190
  4.9.2 Edition Area 195
  4.9.3 Date Area 196
  4.9.4 Physical Description Area 198
  4.9.5 Notes Area 200
  4.9.6 Main Entry Heading 201
4.10 Criticism of AACR2R : Manuscripts 204

CHAPTER 5 - NATIONAL REGISTER OF MANUSCRIPTS

5.1 Introduction 213
5.2 Archives and Manuscripts 214
5.3 Similarities and Differences : Books, Manuscripts and Archives 217
5.4 Computerisation 242
5.5 Government Archives Services 243
5.6 National Register of Manuscripts 246
5.7 Components of Narem Entry and Implications for Design 247
  5.7.1 Depot 249
  5.7.2 Type 250
  5.7.3 Reference 250
  5.7.4 Title 251
CHAPTER 6 - COMPOSITE MODEL

6.1 Introduction 260
6.2 Composite Model Development: Approach taken in study 263
   6.2.1 AACR2R and Modelling 264
   6.2.2 Completeness of Description 265
   6.2.3 Level of Description 265
   6.2.4 Identification 269
   6.2.5 Reporting 269
   6.2.6 General Material Designation 269
   6.2.7 Classification (Shelf Arrangement) 270
6.3 Model Reconciliation: NAREM and AACR2R 270
6.4 Mapping Attributes onto Relational Tables 279
6.5 Denormalisation 282
6.6 Indexes and Indexing 288
6.7 Variable Length Fields and Clipper 304
6.8 Fine-tuning using Memo Fields 306
6.9 Algorithms in Clipper 311
CHAPTER 7 - PROGRAMMING AND REPORT GENERATION

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1 Introduction</td>
<td>318</td>
</tr>
<tr>
<td>7.2 Requirements Definition</td>
<td>321</td>
</tr>
<tr>
<td>7.3 Validation and Control</td>
<td>322</td>
</tr>
<tr>
<td>7.4 Screen Design</td>
<td>327</td>
</tr>
<tr>
<td>7.5 Approach to Programming</td>
<td>331</td>
</tr>
<tr>
<td>7.5.1 Input Program</td>
<td>334</td>
</tr>
<tr>
<td>7.5.2 Update Program</td>
<td>348</td>
</tr>
<tr>
<td>7.5.3 Deletion and Packing</td>
<td>350</td>
</tr>
<tr>
<td>7.6 Reports</td>
<td>351</td>
</tr>
<tr>
<td>7.6.1 NAREM Format</td>
<td>354</td>
</tr>
<tr>
<td>7.6.2 AACR2R Format</td>
<td>359</td>
</tr>
</tbody>
</table>

CHAPTER 8 - FINDINGS, RECOMMENDATIONS AND CONCLUSION

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1 Findings</td>
<td>366</td>
</tr>
<tr>
<td>8.2 Recommendations</td>
<td>385</td>
</tr>
</tbody>
</table>

9. APPENDICES 395
10. REFERENCES 478
LIST OF ABBREVIATIONS

AACR2  - Anglo-American Cataloguing Rules, Second Edition
AACR2R - Anglo-American Cataloguing Rules, Second Edition, Revised
ALA    - American Library Association
APPM   - Archives, personal papers and manuscripts
ASCII  - American Standard Code for Information Interchange
BM     - Bibliographic Model
BRN    - Bibliographic Record Number
DBA    - Database Administrator
DBMS   - Database Management System
DDL    - Data Definition Language
DML    - Data Manipulation Language
DOS    - Disk Operating System
E-R    - Entity-Relationship
GAS    - Government Archives Services
GMD    - General Material Designation
ISAM   - Indexed Sequential File Access Method
ISBD   - International Standard Bibliographic Description
MAR    - Manuscripts and Archives
MARC   - Machine Readable Cataloguing
MARC:AMC - MARC : Archival and Manuscripts Control
NAREM  - National Register of Manuscripts
NEH    - National Endowment for the Humanities
OCLC   - Online Cataloguing Library Centre
PDA    - Physical Description Area
RLIN   - Research Libraries Information Network
SQL    - Structured Query Language
STAIRS - Storage and Information Retrieval System
### LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Illustration Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Data Independence</td>
<td>38</td>
</tr>
<tr>
<td>2.2 Components of a DBMS</td>
<td>40</td>
</tr>
<tr>
<td>2.3 Bachman Diagram of Issues System</td>
<td>47</td>
</tr>
<tr>
<td>2.4 Two-Dimensional Table for Supplier</td>
<td>48</td>
</tr>
<tr>
<td>2.5 Relations A and B</td>
<td>54</td>
</tr>
<tr>
<td>2.6 Relational Operators</td>
<td>55</td>
</tr>
<tr>
<td>2.7 Bib, Item and Borrower Record Definitions</td>
<td>62</td>
</tr>
<tr>
<td>2.8 Field List from Bib View using Report Form Bib</td>
<td>72</td>
</tr>
<tr>
<td>2.9a-b Select and Project Operations</td>
<td>74</td>
</tr>
<tr>
<td>2.10 Screen for Borrower Records</td>
<td>79</td>
</tr>
<tr>
<td>2.11 Report Form for Borrower Records</td>
<td>80</td>
</tr>
<tr>
<td>2.12 Overdue and Over Subroutines</td>
<td>83</td>
</tr>
<tr>
<td>2.13 Report of Overdue Books</td>
<td>87</td>
</tr>
<tr>
<td>3.1 Design Process</td>
<td>99</td>
</tr>
<tr>
<td>3.2 Employee and Salesman Entities</td>
<td>123</td>
</tr>
<tr>
<td>3.3 ISA Representations</td>
<td>124</td>
</tr>
<tr>
<td>3.4 E-R Model of Issues System</td>
<td>125</td>
</tr>
<tr>
<td>3.5 Files showing many-to-many Relationships</td>
<td>130</td>
</tr>
<tr>
<td>3.6 Example of Index</td>
<td>138</td>
</tr>
<tr>
<td>3.7 Inverted File Organisation</td>
<td>139</td>
</tr>
<tr>
<td>4.1 Typical Catalogue Entry</td>
<td>162</td>
</tr>
<tr>
<td>4.2 Multi-level Description</td>
<td>166</td>
</tr>
<tr>
<td>4.3a Alternate Entry</td>
<td>169</td>
</tr>
<tr>
<td>4.3b Added Entry using Main Entry Principle</td>
<td>169</td>
</tr>
<tr>
<td>4.3c Added Entry for Alternate Entry</td>
<td>169</td>
</tr>
<tr>
<td>4.4 ISBD format</td>
<td>176</td>
</tr>
<tr>
<td>4.5 First Level ISBD</td>
<td>179</td>
</tr>
<tr>
<td>4.6 Second Level ISBD</td>
<td>179</td>
</tr>
<tr>
<td>4.7 Individual and Collections approach using ISBD</td>
<td>183</td>
</tr>
<tr>
<td>4.8 2nd Level ISBD for Manuscripts</td>
<td>184</td>
</tr>
<tr>
<td>4.9 Characteristic Entity</td>
<td>188</td>
</tr>
<tr>
<td>4.10 E-R Model of Title Area Elements</td>
<td>195</td>
</tr>
<tr>
<td>4.11 E-R Model of Edition Area Elements</td>
<td>196</td>
</tr>
<tr>
<td>4.12 E-R Model of Date Area Elements</td>
<td>198</td>
</tr>
<tr>
<td>4.13 E-R Model of PDA Elements</td>
<td>199</td>
</tr>
<tr>
<td>4.14 E-R Model of Note Area Elements</td>
<td>201</td>
</tr>
<tr>
<td>4.15 Model showing Main Entry Heading in BM</td>
<td>202</td>
</tr>
<tr>
<td>4.16 E-R Model for Manuscripts, Level 2</td>
<td>203</td>
</tr>
<tr>
<td>5.1 NAREM Entry</td>
<td>248</td>
</tr>
<tr>
<td>5.2 Remarks Paragraph</td>
<td>253</td>
</tr>
<tr>
<td>5.3 E-R Model of NAREM Elements</td>
<td>254</td>
</tr>
</tbody>
</table>
6.1 BM (part) including Reference, Class and BRN
6.2a BM showing Depot
6.2b Depot Entity
6.3 Notes included in the BM
6.4 E-R Diagram of BM
6.5 Table1
6.6a-h Title, Resp, Edition, Resp2, date, Physdesc, Notel and Depot Tables
6.7 Table1 (Denormalised)
6.8 Procedure Comain
6.9 Main Entry Heading and Title Sort
6.10 Table1 (After including Index Fields)
6.11 Sub-dictionary definition for Mem field
6.12 Memo field (Mem) structure within Table1
6.13 Table1 (Final)
6.14 Subroutine Connote
6.15 Subroutine Commem
6.16 Subroutine Ucommem
6.17 Subroutine Uconnote

7.1 Depot Authority File
7.2 Fixed length Field Screen Format
7.3 Screen Layout
7.4 Aprog Options
7.5a Main Entry Heading Screen Options
7.5b-e Names, Place, Corporate and Uniform Title Screens
7.6 General Screen (Utility Screen)
7.7 Notes Screen
7.8 Update Options
7.9 Worksheet Format
7.10a-c NAREM Entries
7.11a-g AACR2R Entries
DECLARATION

I hereby declare that this dissertation is my own unaide work and that it has not been submitted for a Master’s or any other degree at any other university.

K. Chetty
CHAPTER 1
INTRODUCTION

1.1 Introduction
Traditionally, cataloguing of manuscripts have followed two divergent approaches, one according to the archival tradition and another using a 'library' approach. Proponents of the archival tradition have argued that manuscripts are not books and thereby deserve special treatment in terms of their arrangement, administrative control, description and tools used to access a manuscript collection. Early manuscripts were catalogued along library principles. Each item was catalogued separately. The sheer bulk of manuscript collections gave way to bibliographic control at the collection level. Out of this rose the archival tradition with its own set of principles and practices.

Those following the library approach argue that manuscripts are just one form of library materials and share similar characteristics to books and should not be treated differently. Hence manuscripts are catalogued and classified similarly to books, with a few adaptations. These two approaches gave rise to two sets of 'cataloguing codes', namely, Manual for Archival Descrip-
tion for archives and a multi-media library cataloguing code, namely, the Anglo-American Cataloguing Rules, second revised edition (AACR2R).

With more and more archival and manuscript records appearing in bibliographic databases such as the OCLC (Online Cataloguing Library Centre), manuscript cataloguing using the AACR2R has become a reality overseas. One of the immediate implications of such developments for institutions in South Africa, is the possibility of developing a single database on books and manuscripts using the AACR2R with possible extensions to accommodate variant practices.

In South Africa, the State Library maintains a national register of manuscripts (NAREM) on an online database. Individual institutions could up to now join the NAREM project by submitting entries in pre-defined worksheets or data forms. These entries are captured in the NAREM database and made available online throughout the country.

Since NAREM does not use the AACR2R for cataloguing, this poses a problem for institutions that have manuscripts as part of their holdings and desire to treat such material
as only one form of material in an integrated collection consisting of serials, books, audio-visual material, maps and manuscripts. The descriptive details for each entry required by the AACR2R and NAREM differ significantly, and any institution wanting to participate in the NAREM project must take cognizance of this.

NAREM has its own set of standards and specifications not necessarily consistent and compatible with the AACR2R. When exchanging data it is important to ascertain what data elements are required for a given entry. The requirements of local institutions must be compatible within the framework of national requirements, if exchange is to take place.

A typical NAREM entry consists of the depot, reference, type of accession, title, remarks, summary, opening and closing dates that reflect the documents content. The AACR2R makes provision for a main entry heading, title and statement of responsibility, edition, date, imprint, and notes. These elements do not necessarily correspond to the data required by NAREM, yet there are sufficient areas of commonality and differences in both types of descriptions. The title for both the descriptions are more or less similar and are compatible if the AACR2R
code is used. However, the depot, reference and remarks paragraphs need interpretation according to the AACR2R rules. For instance, the remarks paragraph in an NAREM entry has a hybrid of notes and physical description details and this needs translation into their equivalents in the AACR2R format, where possible. The depot and reference number have no place in a AACR2R description. Further, NAREM entries do not have an edition statement or a main entry heading. On the basis of these similarities and differences it is not possible to use any one set of requirements to accommodate the other. The solution is to develop a composite model to satisfy both requirements. Both type of entries can be constructed from the composite model.

1.2 Computers, Databases and Cataloguing Codes

Cataloguing is an important aspect of a library’s activities. As such it must be approached with proper planning to ensure that the final product is both efficient and effective. One of the main objectives of cataloguing is to produce a list or catalogue of entries. ‘A catalogue is a list of, and index to, a collection of materials’. Cataloguing is ‘the art of describing and listing material in such a way as to make it easy as pos-
sible to discover the nature and extent of what is available and, if appropriate, where this material may be located or obtained’ (Hunter, 1985:1).

Computerisation of the catalogue has many benefits. It can improve efficiency, services and increase cooperation with other libraries and information services (Hunter, 1985:3). Computer-based information retrieval systems offer more convenient, more flexible and more comprehensive retrieval than manual systems (Rowley, 1980:8). Computer programs can cross-reference files and data easily but would be time consuming when done manually (Liskin, 1987:12). Manual catalogues have only a limited capacity to relate data.

Different kinds of programs are available today. A Database Management System (DBMS) is a special purpose program that enables the creation of record structures, screens and reports. Through a series of simple and complex programs redundancy and duplication can be eliminated or controlled. A DBMS is capable of integrating the various files in a system and co-ordinate different activities.
DBMS define the record structures in terms of fields and subfields. This formal structure is based on an analysis of user requirements and the resulting catalogue entries that result from these requirements. Formal structures are capable of enforcing standards, maintaining or ensuring consistency, and ensuring compatibility. By using appropriate programs, a standard description of a catalogue entry can be produced in varying formats, provided that sufficient specificity has been included in the design of the database. The capability of programs to manipulate bibliographic elements gives the appearance of adherence to cataloguing codes when in fact it is ordering the elements in ways reflecting well defined requirements. Provided that the requirements are well defined, computerisation facilitates the entry of data once and data output in one of several different formats. In this sense, the computer if properly utilised, can afford a truly significant advance in controlling the catalogue, a model of which is represented in the formal structures as defined in the DBMS. Ideally, only one composite structure or model should exist for different user requirements.
With the emergence of micro-computerised database systems small applications have become viable. Dbase III+ is one such example. The study used the potential of Dbase III+ to develop and test a composite model to satisfy both the requirements of the AACR2R and NAREM.

Theoretical framework
It is possible to develop a composite model in Dbase III+ that would satisfy both the requirements of the AACR2R and NAREM. The study set out to develop and test a model database.

1.3 Aims and Objectives of the Study

1. To develop a composite database model that is AACR2R and NAREM compatible.

2. To investigate the kinds of options and alternatives available in developing a composite model using the capabilities of Dbase III+.

3. To investigate any advantage(s) and limitations of the AACR2R and the NAREM formats in their ability to accommodate a wider format that is AACR2R and NAREM compatible.
4. To bridge the existing gap between two apparently divergent traditions, the library and information services on one hand and the Archive and Manuscript tradition on the other.

1.4 Literature Review

Numerous papers have dealt with a discussion of the AACR2R, the ISBD and need for standardisation in cataloguing (French, 1981; Gorman 1978b; Hoffman, 1983; Milcetic, 1982; Thomas, 1987; Rowley, 1989; Svenonius, 1989; Taylor, 1988; Tucker, 1990; Wajenberg, 1990; Boll, 1990). However, little attention has been paid to the cataloguing of manuscripts using the AACR2R. This is probably due to manuscripts being treated as less important than other materials and that manuscripts are treated as another form of library material alongside books, maps, music sheets, audio-visual material and sound recordings. Works on the archival tradition, on the other hand, have emphasised the principles and management of archives and manuscripts (Holmes, 1964; Olivier, 1983; Schellenberg, 1956). However, similarities and differences between the two traditions were highlighted (Fox, 1990; Schellenberg, 1956; Hensen, 1983; Roe, 1990; Hesselager, 1984; Berner, 1965; Berning, 1971).
Schellenberg, (Duminy, 1986:23) remarks that manuscripts have the same organic quality as archives and justifies their documentation according to archive rather than library techniques. R C Berner and L Brubacker pointed out that 'the bibliographic characteristics of archives and manuscripts are fundamentally the same' and that archival principles can be applied to general manuscripts (Berning, 1971:8). Hensen (1983:1) agrees that with appropriate modifications library based descriptive techniques can be applied to manuscripts and archives. Hence modifications of the AACR2R in manuals such as the 'Archives, personal papers and manuscripts: a cataloging manual for archival repositories, historical societies and manuscript libraries', have emerged (Hensen, 1989).

For a computerised environment, the MARC:AMC (Machine readable cataloguing: archives and manuscripts control) features prominently (Madden, 1991; Gertz & Stout, 1989; Gilliland, 1988; Hensen, 1988). In other instances, computerisation revolved around codes such as the Manual of Archival Description (Cook, 1986) and other computerised efforts (Cook, 1980). With the introduction of computerisation in South Africa, the NAREM project came into being and many papers (Ferreira, 1991; Ferreira, 1983;
NAREM, 1985; Olivier, 1983; Smith, 1978) on the nature and automation of manuscripts with respect to the project have emerged. Merrett (1990) in particular has evaluated the computerised system of the Government Archives Services and compared it with the Urica System for libraries.

The machine readable form of the catalogue has stimulated discussion on various standards like the MARC (Machine Readable Cataloguing), ISBD formats and various other formats required for the exchange of data between institutions at the local, national and international level (Avram, 1974; Crawford, 1982; Knapp, 1968; Thomas, 1987; Gredley, 1990; McIlwaine, 1991). Formats point to the way files are designed and organised. An aspect of computer science involves a step by step approach to designing files by adopting file designs techniques of normalisation (Caswell, 1984; McMurdo, 1982) and entity modelling (McFadden & Hoffer, 1991; Ullman, 1982; Atre, 1980). Once the initial data is modelled then the resultant model is mapped onto a relational, hierarchical or network model. These activities fall within the ambit of database design.
With the introduction of computers, database design was considered as part of Information Science (Caswell, 1984; Agosti, 1989; Azubuike, 1989; Pangalos, 1989). Concepts of relevance to database design were considered as an information science related activity (McMurdo, 1982). Studies have focused on data relationships and file structures of bibliographic records (Koenig, 1985; Fosdick, 1981; Rowley, 1980; Gredley, 1990). In particular, studies in library science have concentrated on the different kinds of databases and in particular the relational database (Brooks, 1985; Agosti, 1989; Pangalos, 1989) thereby bringing databases within the field of library science.

Some studies have been more specific revolving around a discussion of database management systems and the design of personal databases using Dbase II/III+ and Dbase IV (Mcfadden & Hoffer, 1991; Armstrong, 1984; Sullivan, 1985; Bordwell, 1984; Chun, 1987; Hayman, 1990). Chun (1987) for instance, describes the use of Dbase to set up a database for the storage of bibliographic information. Beiser (1987) contains numerous applications of Dbase III+ for different activities in the library field, namely, generating catalogues and setting up different systems for serials, interlibrary loans and acquisitions.
This brief survey indicated that many of the concepts that were to be dealt with in the study were treated in a fragmented manner. Hence, the aim of the study was to combine issues relating to cataloguing, standardisation, computerisation, manuscript cataloguing, data modelling and database design into one integrated whole. The ultimate aim was to illustrate how all these aspects could contribute to an integrated approach in designing a database model for manuscripts based on entries as outlined by the AACR2R and NAREM. The final objective was to produce reports for NAREM and AACR2R from the same composite model so that institutions could report to NAREM while retaining existing library cataloguing practices.

The fundamental reality is that both library and archives keep manuscripts. The study therefore set out to demonstrate that an integrated database of the AACR2R and NAREM was possible. The conceptual model arising from the study could then be used for the many existing sophisticated computerised library systems such as URICA, SLS Plus and Dobis, that are in use, today.

The GAS computerised system is not sophisticated enough\(^1\) to handle the requirements of the AACR2R but library systems could be extended and modified to include NAREM ele-
ments. Today many library-based systems adopt the AACR2R or some earlier version of the code and they could benefit from the study once solutions are advanced. The MARC:AMC has not received wide acceptance and usage in South Africa. Hence the MARC format, if used, can be extended to include elements of the MARC:AMC and where elements are peculiar to NAREM only, additional provisions in existing databases could be made.

Merely adding new fields to existing databases is not adequate as the conceptual aspects need to be understood first. The similarities and differences of both the AACR2R and NAREM requirements highlighted in the study, against the background on the nature and usage of the respective elements would lead to the proper placement and creation of fields in existing databases. The use of Dbase in the study serves two purposes, firstly, to test and validate the conceptual model and secondly, to pave the way for micro-computer based systems for small applications for manuscripts using the AACR2R and NAREM. With respect to the latter, the application developed in the study is complete. However, extensions to the model is required for other forms of library materials. Where other formats and codes are being used then the constraints posed by the requirements must be considered.
before any extensions can be made to such systems and up to now except for Merrett (1990), published literature incorporating NAREM into existing systems is largely absent.

1.5 Method of Developing and Testing the Composite Model

Two basic methods of developing and testing the data were employed in the study, that of database design and programming. These aspects are dealt with below. However, a discussion of the sample used in this study follows immediately.

SAMPLE

The sample of entries used in the study were taken from different sources to illustrate divergent practices. The sample consisted of 100 entries divided according to the following sources:

<table>
<thead>
<tr>
<th>Code</th>
<th>Source</th>
<th>Number in Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uct</td>
<td>University of Cape Town Libraries Libraries</td>
<td>10</td>
</tr>
<tr>
<td>Unisa</td>
<td>University of South Africa</td>
<td>10</td>
</tr>
<tr>
<td>KAB</td>
<td>Cape Archives Depot</td>
<td>10</td>
</tr>
<tr>
<td>APPM</td>
<td>Archives, personal papers and manuscripts</td>
<td>20</td>
</tr>
<tr>
<td>KCM</td>
<td>Killie Campbell Africana Library</td>
<td>18</td>
</tr>
<tr>
<td>UDWT</td>
<td>Theses at University of Durban-Westville</td>
<td>8</td>
</tr>
<tr>
<td>UDW</td>
<td>Documentation Centre (UDW)</td>
<td>24</td>
</tr>
</tbody>
</table>
Examples 1 to 30 were chosen to test existing NAREM entries. The examples are found in the following respective publications 'Guide to the manuscripts in the University of Cape Town Libraries', pages 1-5, 'Guide to the archival collections in the Unisa Documentation Centre for African Studies', pages 2-6 and 'Guide to Accessions in the Cape Archives Depot, Cape Town', pages 98-100. Each of the three categories differ. Examples 1 to 10 illustrate NAREM entries of type manuscripts, examples 11 to 20 NAREM entries of type accessions while examples 21 to 30 illustrate examples taken from an archive depot, of type accessions. The examples were taken from a documentation centre (UNISA), an archive depot (KAB) and a network of libraries (UCT). All entries were limited to collections.

The examples taken from the APPM (Hensen, 1983), pages 29-43, are significant in that they represent both item level and collection level cataloguing using an adaptation of the AACR2. In addition, the examples have many features, such as, main entry headings, relationship complexity notes and incorporated data from manuscript and archival collections from more than one institution. Converting these records into the NAREM format posed a challenge and in the process would highlight any drawbacks of
limiting the design to just NAREM and AACR2R examples. In this regard, special attention was paid to the extensive notes provided for in the APPM.

The examples taken from the Killie Campbell Africana Library are found in Merrett (1990). The examples illustrate the use of the APPM and highlight multi-level cataloguing. The Documentation Centre examples, on the other hand, are NAREM and AACR2R compatible and illustrates multi-level cataloguing for collections, individual items and intermediate levels as well. The records were specially prepared from the Centre’s card catalogue and the publication ‘A bibliography on Indians in South Africa: a guide to materials at the Documentation Centre / compiled by K Chetty’, section A. Theses are important items in research institutions and these have been selected from the University of Durban-Westville Library. The examples selected conform to the AACR2R.

It was hoped that by selecting a hybrid of examples, the study would go beyond just ascertaining the compatibility of NAREM and AACR2R but in its wake would point the way to a composite model that encompassed a wider archive and manuscript environment.
Database Design Methodology

When developing a composite model for a particular DBMS established techniques and methods should be used. Database design provides the methodology for the analysis and development of a database model.

Database design is the process of developing database structures from user requirements for data. It starts with requirements definition, which identifies user needs (present and future) for data. It then proceeds by translating these user requirements first into a logical, then a physical database design. The resulting design must satisfy user needs in terms of completeness, integrity, performance constraints, and other factors (McFadden & Hoffer, 1991:167).

From the above definition, the basic steps in designing a database are:

1. Define the user requirements.
2. Develop a logical model from the user requirements.
3. Choose a particular DBMS and map the entities and relationships from the logical model into equivalent files.
4. Fine-tune the database model (physical design) by choosing appropriate indexing techniques, files etc.
The above steps have been covered by one or more chapters in this study. The characteristics of Database Management Systems (DBMS) is dealt with in chapter 2. Particular attention was given to Dbase III+ as an example of a relational model. The various steps outlined in the design process is examined in chapter 3. Modelling techniques and concepts, for example, data normalisation and E-R modelling, are examined. The former emphasises elements of a good design while the latter serves as a tool to representing data graphically.

In chapters 4 and 5, the two user requirements that correspond to the two approaches, were discussed and appropriate E-R models were developed. Chapter 4 considers the library approach to the cataloguing of manuscripts as outlined in chapter 4 of the AACR2R while chapter 5 deals with the descriptive elements as required in a typical NAREM entry.

The individual E-R models developed in chapters 4 and 5 were combined into a composite model in chapter 6. The model at this stage, is, however, incomplete, as key issues, namely, that of multi-level descriptions or analysis and classification were omitted. These aspects
including other minor aspects were subsequently included in the model to yield a composite Bibliographic Model (BM). Thereafter the composite model, was translated into appropriate Dbase tables. Important aspects relating to record structures, authority file creation and indexing formed the subject of the chapter. The process of denormalisation was applied to fine-tune the database.

The development of suitable input, update, deletion and report programs were considered in chapter 7. Subsequently, two reports, an AACR2R and NAREM report were produced from the composite model. The chapter was followed by a series of conclusions and recommendations in chapter 8. Chapters 3-7 in particular, reflect the step by step approach to the design, development, programming and reporting with respect to the composite model. Programming is a technique and art itself requiring thorough organisation and planning and is considered below.

**Dbase Programming Language**

Programming was the second important technique used to test the composite model. The model was tested via a series of programs written in Dbase III+. Where the language was found to be inadequate, Clipper was used. For
example Dbase III+ is unable to accommodate a variable of 32k and at least 1.5k is required for NAREM records that have particularly long abstracts. In addition, the extended Clipper environment, enables memo fields to be searched. Additions and updates to memo fields were made possible by using the standard 'replace' commands used for character, numeric and logical fields.

Even though Clipper is a stand-alone language it was used exclusively as a Dbase III+ compiler with language enhancements. By focusing on Dbase III+ instead of Clipper, the fundamental and basic aspects of this popular software were explored, features which are included in versions such as DRXL and Foxbase. A discussion of Clipper as the basic language in the study, was therefore excluded to make way for a wider family of related languages while simultaneously bringing to the fore enhanced features of Clipper.

The Programming Process and Dbase Programming
Once the information needs were ascertained, the input, processing, output, storage and control requirements of the proposed system were determined. Specifications regarding the output required, the input available, the processing that was required and the control procedures
that would be needed, were defined (O’Brien, 1985:283-285). Thereafter, computer programs were developed to meet the information needs.

Algorithms or a set of rules and instructions that specify the operations were outlined and coded into a set of Dbase programming language statements. After the program was coded, it was verified or debugged. This stage involved checking, testing and correcting the program statements, the program logic and each one of the modules or subroutines developed. Only when the programs demonstrated that they could work correctly together, were the programs considered to be fully tested (O’Brien, 1985:299-303).

To ensure that the programs worked correctly, test data from real life situations were used. Use of real data allows the checking of the accuracy of the data and the processes they were involved in. The test data was carefully selected to simulate different conditions that may have arisen during processing. Where the sample did not reflect particular conditions, artificial records were created to test conditions not reflected in the sample.
The test data served to ascertain whether the programs could handle exceptions and errors as well as more normal forms of data. The test data was used to pin-point problems related to the file structures; any fields that may have been overlooked; length of the fields in the file or any variations in two fields from different files that were used as common keys (Liskin, 1987:44-5). Thus testing served to confirm whether the file structures could accommodate the required data and any missing links or relationships among the data files would be discovered. In addition, testing guaranteed that the necessary tasks could be carried out successfully using Dbase III+ (Liskin, 1987: 43-4) and hence provided a means to validate the model and reveal any design flaws.

The success of a model is measured by the extent to which the database can represent the data and produce appropriate reports to reflect NAREM and AACR2R requirements. When testing for the AACR2R, special emphasis was given to whether a typical entry in AACR2R, can be successfully constructed from the composite model using hanging indentations for both the main and alternate entry systems. With regards to the NAREM format, emphasis was given to the ability of the system to produce entries as specified by NAREM.
1.6 Scope and Limitations of Study

The study focused on the construction of AACR2R and NAREM entries. The former includes the main entry heading and the ISBD format. While cognizance was taken of the rules for the construction of headings, such emphasis was given in so far as they had implications for database design. The rules point to various elements such as the General Material Designation, title, authors or publishers, that may be included in the description. These elements and rules were considered if they had implications for the existence, order and optional nature of the elements which are of relevance in designing a system. Similar consideration were given to the NAREM elements.

The study was limited to the investigation of the feasibility of a new system to accommodate manuscript descriptive elements and was not intended to be a study of information retrieval systems. Therefore issues such as performance, use of the various access strategies for retrieval, for instance, were not considered.
The study includes only manuscripts as defined in chapter 4 of the AACR2R and excludes other materials, for example, maps, music sheets and unpublished material. Hence the descriptive areas were limited to those defined in chapter 4 of the AACR2R.

1.7 Reference Technique

All references are listed alphabetically by main heading which may be an author, title or corporate body. The author(s) (up to three) are cited followed by the year and the page(s), for example (McFadden & Hoffer, 1991:14-15). Where the authors formed part of the syntax of a sentence being used, only the year and pages were cited within brackets, for example ‘According to Date (1981:4)…’. Whenever mention was made of more than one publication for an author for a particular year, the dates were extended to include the alphabets a, b, c etc. for the occurrence of each publication in that year. The alphabets were assigned in the order the item appears in the bibliography. Examples: (Date, 1981a:4), (Date, 1981b:17-18).

Some headings were very long for inclusion in a reference and a system of acronyms was devised to reference the publications they represent. Some of acronyms used in the
study are SAMARC, NAREM, GAS and AACR2R and this usually followed the heading, if the acronym does not already appear at the beginning of the heading, for example 'National Register of Manuscripts (NAREM). Manuscript collections and NAREM....'. The publication is cited as (NAREM, 1985:5) throughout the study.

NOTES

1. See criticism of the computerised system of the Government Archives Services and in particular NAREM, discussed in chapter 5 of this thesis.

2. Both concepts are normally used to compliment each other.

3. Both aspects are dealt with in chapters 5 and 6 of this study.
CHAPTER 2
DATABASE MANAGEMENT SYSTEMS

2.1 Introduction
The heart of any information system is its stored data. An enterprise deals with information on one or more aspects of its operational environment. For example, a library system consists of a cataloguing subsystem, a circulation control subsystem, a serials subsystem as well as a management subsystem to provide statistics for management on student usage or number of books ordered. Data is central to a library or any other system, the data itself serving as input or forms part of the data being output during the course of a system's operations and functions.

In a computerised environment, computers are able to process vast amount of data more speedily and efficiently than manual methods (Hunter, 1985:2). Database Management Systems (DBMS) provide the environment for the processing and manipulation of data in a computerised environment. A DBMS is 'a generalized set of computer programs that controls the creation, maintenance, and utilization of the databases and data files of an organization' (O'Brien, 1985:G-7). However, before data can be
manipulated or processed, it must be organised into a well defined and structured database. 'A database is a collection of similar records, with relationships between the records' (Rowley, 1980:49). Azubuike (1989:244) places a further requirement on a database. He states that the data items should be processable by one or more application systems. Unlike the master record approach\(^1\), where files are developed according to the needs of specific users, the DBMS approach enables the building of an integrated database.

The concept of DBMS was first used in mainframe computers and later extended to Personal Computers (PC's) using DBMS packages like Dbase and Oracle. The emergence of high-capacity, fast and inexpensive hard disks have made micro-computer-based library catalogues a realistic possibility for small and even not-so-small libraries. Typical PC_DBMS (Personal Computer-based Database Management Systems) include elements like screen painters to design screens; report writers to layout the format of a record; query languages to retrieve data; special procedural and command languages to control the order of processing; a dictionary facility to define data; application generators to develop turnkey systems and a
graphics utility that displays the content of a database in the form of pie charts, line graphs and bar charts (McFadden & Hoffer, 1991:550-1). Many of the features of PC_DBMS are similar to the features of mainframe DBMS, but not necessarily as powerful. PC_DBMS are limited by their processing speed, word size, the size of the database (McFadden & Hoffer, 1991:544-547) and the number of users and applications it can accommodate. In some cases features of mainframe DBMS are missing in PC_DBMS. In addition to providing standard DBMS facilities, other management support tools like spreadsheets and graphic facilities may invariably be part of the PC_DBMS package. PC_DBMS have the advantage in that they can provide a link to mainframes with the same range of commands as their mainframe counterparts, share some of the mainframe processing functions and provide additional management support tools. In addition, PC_DBMS are more portable for use in the home, classroom or laboratory environment (McFadden & Hoffer, 1991:548). Why is the Database approach chosen? What are the advantages of the database approach? The variety of functions to describe and define data, the provision of screen and report interfaces, query facilities as well as
the provision of numerous file handling facilities (for example, erasing, backup, linking data to spreadsheets and wordprocessors etc.) are provided by the DBMS. According to O’Brien (1985:401), a typical DBMS facilitates the creation, interrogation and maintenance of a database. When a database is created, the records, indexes and fields, their data types (character, numeric, logical) are defined. Database interrogation involves accessing the data to support various information processing tasks that require information retrieval and report generation. Database maintenance refers to the addition, deletion, updating, correcting and protection of data in a database.

2.2 Why Use a DBMS
The structure of early computerised files followed closely to the structure of their manual equivalents, that is, files were based on specific user needs and data requirements, which in turn determined the nature of the application. As a result, a master record was developed for a particular activity, keeping all elements relevant to that activity in one file (Caswell, 1984:293). Over time different applications gave rise to further files which often duplicated data in existing files.
To illustrate this point, consider two library applications, one for issuing of books to borrowers (Borrower File) and another for a book reservation system for lecturing staff (Reservation File). Lecturers place books on reserve for use by students. While the Borrower file has details such as the Borrower’s Number, Name, Address and Telephone Numbers, the Reservation file has similar details except that the Borrower number is replaced by a Staff Number. In addition, the Reservation file has a course field to identify the course for which the item is reserved. Furthermore both applications would require item details such as the author, title and classification number. Separate applications mean unnecessary duplication. There is sufficient reason to combine the applications. A common Borrower File with one system of numbering for staff and students with an additional field for course would result in an integrated file. By placing all the bibliographic details in one file, both applications can use the Borrower and Bibliographic files.

Separate applications mean that data items may be defined differently, for example, a title having 100 characters in one application and 200 characters in another. This leads to limited data sharing and a poor enforcement of
standards in the organisation of files. A change in one data item, for example, length from 20 to 30 characters necessitates changes in all appropriate files, thereby increasing program maintenance and developmental costs and hence lower programmer productivity.

Separate files on the other hand, not only lead to duplication but lead to inconsistent data. For example, one lecturer who is also a borrower would have two master records in two different applications with the same details, namely, the patron’s name, address and telephone numbers in both files. If the address and telephone number is changed in one file and not in the other, the files would reflect the old and new addresses, leading to inconsistencies for the same data. There is no guarantee that data in both files would be updated simultaneously as each application in the master record approach operates independently of other applications. Inconsistent data undermine the confidence of users in the integrity of an information system, especially when such inconsistencies are glaring in documents and reports generated by an organisation (McFadden & Hoffer, 1991:14).
DBMS have evolved to solve some of these problems mentioned above. When using a DBMS, data is defined independent of the applications that use them so that all data are controlled centrally and forms a joint asset of all departments (McFadden & Hoffer, 1991:12-13). The database approach offers a number of potential benefits when compared to the master record approach. These benefits include minimal data redundancy; consistency of data; integration of data; sharing of data; enforcement of standards; ease of application development; uniform security and privacy; data accessibility and responsiveness; data independence and reduced program maintenance (McFadden & Hoffer, 1991:21). Benefits are however, inter-related and inter-dependent, for example, standards imply the possibility of sharing and non-redundancy. Hence related characteristics are grouped as shown below.

**Data Integration, Shareability, Non-Redundancy and Consistency**

Using the master record approach for the issues and reservations examples, cited above, each application would define the data elements according to their own requirement. For example, a title of 300 characters in one application and 100 characters in another, the
telephone number having 10 characters in one application and 12 characters in another. The result is a lack of standardisation. Ideally data should be defined only once forming a common pool to be shared by all applications (data integration). A DBMS records data definitions only once in a dictionary or directory instead of each application defining its own data structures and elements (records, fields etc.). As a result, data is organised into one single, logical structure with the logical relationships defined between associated data elements (McFadden & Hoffer, 1991:21). Any changes in one file automatically activate appropriate changes in related files (Rowley, 1980:111).

A total integrated view rather than a piecemeal approach to data and their definitions facilitates data non-redundancy or rather controlled redundancy². The removal of redundancy increases shareability by different users of the same data. A DBMS provides control and access to multiple concurrent views of the same data in a multi-user environment. By eliminating and controlling data redundancy inconsistency is reduced, leading to fewer disagreements in stored values. The DBMS software
integrates the files logically and enforces consistency by updating each occurrence of a data item when any changes occur (McFadden & Hoffer, 1991:21).

**Standardisation, Control, Security and Integrity**

A Data Base Administrator (DBA) is usually vested with the responsibility of database design, data definition and reconciling the requirements of different users into one coherent logical structure. The DBA's duty is to enforce standards, approve and define the data elements, their format and use for different users (McFadden & Hoffer, 1991:22). Once the DBA sets standards for data structures, access and controls, the DBMS enforces these standards during database operations of input, update, deletion and retrieval and provides the necessary security mechanisms to assign, control and remove the rights of access (read, change, insert, delete) of any users to any data item. This includes limiting the rights to sensitive data like medical records and salaries (Cardenas, 1985:92).

A major goal of a DBMS is to maintain control over and preserve the integrity of a database. Integrity refers to the co-ordination of data access by different programs and applications, the update of all values of a data item
and their copies, ensuring consistency of related data items and general validation of the data through consistency checks, input and output editing (Cardenas, 1985:91). Integrity is a DBMS function and excludes inaccuracies by users of the system, for example, improper decisions or entering of data incorrectly.

**Ease of Application Development and Reduced Program Maintenance**

The immediate benefit of integration is savings in developmental costs. Studies have shown that once a database has been designed and implemented, a programmer can code and debug a new application at least two to four times faster than with conventional data files (**Ease of Application Development**). The reason is that the application programmer is no longer saddled with the burden of designing, building and maintaining separate master files for each application (McFadden & Hoffer, 1991:23). Control of data items centrally means that there are fewer changes in programs and subsequently **reduced program maintenance**.
Access Flexibility and Responsiveness

A DBMS provides multiple retrieval paths to data items giving each user greater flexibility in access and the way the system responds to queries. DBMS permit the execution of ad-hoc queries (McFadden & Hoffer, 1991:23). A Structured Query Language (SQL) is able to provide for flexible access to data far beyond that provided by conventional programming languages. In the latter case, separate programs need to be written for each new query. DBMS permit sorting on one or more criteria and access of a subset of data through appropriate selection commands while at the same time restricting access to unauthorised users. It is able to locate data across departmental and artificial boundaries even though the views restrict data access. At a higher level, the integrated view of the data is available to any user with the appropriate access and authorisation privileges.

Data Independence

A DBMS has three views of data; an external (conceptual or logical, for example, Entity-Relationship model) model which is mapped onto an internal model for a particular DBMS, which is also a logical model (second view). The latter model is again mapped onto a physical model (Atre, 1980:84). Use of the three levels of abstraction allows a
degree of 'data independence'. In the model (Fig 2.1) defined by Atre (1980:19), the conceptual and logical models are abstract, but only the physical database actually exists in secondary storage devices such as disks or tapes (Ullman, 1982:6).

Data independence implies independence of, or insulation of application programs from the logical or physical organisation of the data, that is, any changes in the logical or physical organisation do not require changes in the application programs. The ability to change physical data without making obsolete the application programs, is termed, physical data independence while the ability of an application program to continue executing its program correctly in spite of changes in the logical organisation is called logical data independence. Changes are undesirable and imply increased maintenance costs. Fig 2.1 illustrates the areas of data independence.

Physical changes may be due to changes in where the data is located (for example, track x versus track y), changes in physical organisation, for example, sequential versus indexed or inverted file organisations or changes in mode of access (sequential, random, indexed). Logical changes involves changes in field positions, key names, deletion
of fields or changes in record definition, for example, from fixed to variable length records (Cardenas, 1985:13).

![Diagram showing data independence with four applications and three models: conceptual, logical, and internal models.](image)

**Figure 1.7 Two stages of data independence:**

1. **Logical independence.**
2. **Physical independence.**

- **Conceptual model.** The conceptual requirements of individual users are integrated into a single “community” view, called a “conceptual” model.
- **Logical model.** The version of the conceptual model that can be presented to the data base management system is called a “logical” model.
- **Internal model.** The physical model that takes into consideration the distribution of data, access methods, and indexing techniques is called an “internal” model.

**Fig 2.1 Data Independence**

*(Atre, 1980:19)*

In short, a DBMS should accommodate large databases with several users and process data, speedily, efficiently and cheaply. Changes to the system must be made easily with
the least amount of cost involved. Reduced redundancy, increased control, data integrity and security and the ability to interrogate the database in more ways than one, are the primary objectives of a DBMS. In the DBMS approach, databases are designed from an organisation-wide viewpoint. All user requests are routed through the DBMS, which manages the various processes, and effectively serves as a manager at the software level.

2.3 Components of a DBMS
The many functions, for example, definition of data, screens, reports and queries are controlled by a central DBMS engine using one or more subsystems. McFadden & Hoffer (1991:360) defines a DBMS as having a central engine that interacts and interfaces with various subsystems to ensure that requests are controlled and routed to the correct subsystem, thereby providing a suitable environment for the requested functions. The module provides access to the repository and the database and co-ordinates all functions of the DBMS. The DBMS engine receives logical requests for data and from the application, determines the secondary storage locations for these data. Once the data is defined, the DBMS 'knows' where and how to store the data (Ullman, 1982:1)
using its own set of internal rules. Further, the engine provides services such as memory and buffer management, index maintenance and disk management.

Fig 2.2 Components of a DBMS
(McFadden & Hoffer, 1991:361)

McFadden & Hoffer (1991:360-362) identifies seven such subsystems: an Interface Subsystem, a Dictionary Subsystem, a Performance Subsystem, a Data Integrity Subsystem, a Backup and Recovery Subsystem, an
Application Development Subsystem and a Security Management Subsystem. These components are represented in Fig 2.2.

- Interface Subsystem: It consists of a range of languages, for example, a Data Definition Language (DDL) which is used to define database structures such as records, tables, files and views; an interactive query language to interrogate the database and to perform simple updates; one or more programming languages to develop sophisticated applications; an interface to standard third-generation programming languages such as BASIC and COBOL; and a natural language interface that allows users to present requests in free-form English statements. A forms interface, consists of a screen oriented form. This option requires making certain decisions or filling in the blanks. It can be used to add, update, delete and retrieve data, or even define and create files as in Dbase III+. Also included, is a graphics interface, for example, Query-by-Example, in which case the system displays a skeleton table and requests are made by filling in the table. All other subsystems, that is, the dictionary subsystem, the performance management subsystem, the data integrity

41
management subsystem, the backup and recovery subsystem, application development subsystem and security management subsystem interface with this subsystem.

- Dictionary Subsystem: This subsystem controls and manages the repository with commands to add, update and delete definitions of items, records, files, relationships and other elements of the dictionary. The dictionary subsystem contains its own controls and checks for the various elements, for example, checking duplicate file names within a database or duplicate fields within a file. The dictionary is an essential database planning document that must be completed before database automation can begin (Brooks, 1985:12-13).

- Performance Subsystem: Provides facilities to improve and measure database performance, for example, queries can be optimised to minimise response times. A DBMS has utilities to reorganise and restructure the database to improve performance.

- Integrity Subsystem: The subsystem ensures the integrity of data within the record, between records and during usage of records by more than one user. Within the record, data items within restricted ranges, for
instance, are controlled. Special integrity checks are performed, for example, a record does not point to a non-existent record in a database. This generates an error condition. In a multi-user environment, the activities must be synchronised and controlled, for example, whole records and files are locked out from users during the production of a statistical report to prevent inconsistencies due to the database being updated while the report is being produced.

- Backup and Recovery Subsystem: This facility records and logs any changes to a database. Many systems today have roll-back /roll-forward facilities to recover a database. Changes and transactions logged in a transaction log and backup files are used to restore a database and to bring it to a desirable and most up to date state in the event of any failures. Failures may be due to any number of causes, for example, disk crash, power failures, database damage, human or program errors.

- Application Development Subsystem: Provides facilities for end users and programmers to develop database applications. An application consists of one or more programs, files, indexes, screens and reports (called
objects) which are linked together by menus to perform a set of related information processing tasks (McFadden & Hoffer, 1991:629).

- Security Subsystem: Provides facilities to protect and control access to a database and repository. For instance, a query can retrieve sensitive data such as salaries and medical information on an employee. It is not desirable that a user not entitled to view such information, gain access to it. For small applications and single user systems, provided adequate backups are made, such conditions are tolerable but not so in a multi-user environment where access to unauthorised data may be made easier through a lack of security.

A DBMS is truly a manager, leaving specialised tasks to various processors, subsystems and utilities. By managing and relating the various components together and providing access to only authorised users at different levels of access privileges, proper security, control and integrity is maintained (Ullman, 1982:4).
2.4 DBMS Models

There are three types of DBMS, the hierarchical, network and relational, each having its own approach to modelling data (Koenig, 1985:249)\(^3\). The primary objectives of a data model are:

- To serve as a basis for database system architecture;
- To serve as a tool for checking the correctness of specific database system implementations;
- To provide a basis for the development of database design techniques;
- To allow functional requirements and performance requirements to be separately addressed (physical data independence);
- To allow individual and community requirements to be separately addressed (logical data independence);
- To provide a yardstick for the evaluation and comparison of specific database systems;
- To serve as a vehicle for education and research into various aspects of database management (Date, 1983a:186).
Dbase III+ is an example of the relational model. Accordingly, the relational model would receive a detailed treatment. Before embarking on a discussion of the relational model, an example from a library issuing subsystem is considered so that discussion can revolve around it.

Example
An issues system consists of three records, a bibliographic record (Bib) consisting of a Bibliographic Record Number (Brn), a title (Title), author (Author), classification number (Class) and a note (Note). For each Bib record there are many items (Item records) for each copy of a book and is identified uniquely by an Accession Number (Acc_No). Additional fields for the Item record are the price (Price) for each item, due-date (Ddate) and overdue status (Od). The Od status is reset (relative to a given date) whenever a request is made to print all overdue books. The Borrower status comprises of the following user categories: 1 = Undergraduate; 2 = Post-Graduate; 3 = Staff (including lecturing staff). These determine the loan period, namely, 1 = 14 days, 2 = 30 days and 3 = 3 months. The Od is set to true if the item is overdue otherwise it is set to false (logical field).
Full details of the Borrower are recorded in the Borrower file. Each record has a Borrower Number (Borwr_No), name (Name), address (Add), telephone number (Tel) and status (Status). One borrower may borrow more than one book/item. There is a one-to-many relationship between the Borrower and Item records. The entire system is modelled as in Fig 2.3 using a Bachman Diagram. In the diagram each record is shown in a rectangular box linked by lines expressing relationships. The crows foot indicates a 'many' relationship while the straight line represents a 'one' relationship.

Fig 2.3 Bachman Diagram of Issues System
2.4.1 Relational Model

A relational model is concerned with three aspects of data, its structure, manipulation and data integrity (Date, 1983b:249).

<table>
<thead>
<tr>
<th>S#</th>
<th>SNAME</th>
<th>STATUS</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Smith</td>
<td>20</td>
<td>London</td>
</tr>
<tr>
<td>S2</td>
<td>Jones</td>
<td>10</td>
<td>Paris</td>
</tr>
<tr>
<td>S3</td>
<td>Blake</td>
<td>30</td>
<td>Paris</td>
</tr>
<tr>
<td>S4</td>
<td>Clark</td>
<td>20</td>
<td>London</td>
</tr>
<tr>
<td>S5</td>
<td>Adams</td>
<td>30</td>
<td>Athens</td>
</tr>
</tbody>
</table>

**Fig 2.4 Two-Dimensional Table for Supplier**

(Date, 1981:88)

2.4.1.1 Structure

'The basic building block of a relational database is the notion of a flat table, a two dimensional rectangle, in which the rows represent entities and the columns the attributes' (Koenig, 1985:250). Fig 2.4 illustrates a two dimensional table for a supplier, providing the supplier number (S1, S2 etc.), the name (Sname), the status (Status) and city (City). The table is referred to as a relation. The rows of the table are referred to as tuples. A relation is a set and sets are not ordered. Any ordering is system defined, for example, by Borrower number sequence in the Borrower file. In the study, the
terms 'tuple' and 'row' or 'record' shall be used interchangeably. Likewise, columns are usually referred to as attributes or fields. Again these terms will be used interchangeably (Date, 1981:65).

A domain is the possible set of values for an attribute. It is a pool of values from which the actual values appearing in a given column are drawn (Date, 1981:65), for example, the domain for cities in a country are the set of alphabetic characters while the domain for the borrower number are the set of integers greater than or equal to zero.

Each tuple is identified by a primary key which may not contain a null value. Since sets do not contain duplicate elements, each tuple of a given relation is unique with respect to that relation. Hence, at the least, the combination of all attributes uniquely identifies a tuple, however, in practice, some lesser combination of attributes is normally sufficient (Date, 1981:88). Although several different attributes (called candidate keys) might serve as the primary key, only one combination of attributes is chosen, other keys are
called alternate keys. When a key is composed of two or more attributes, it is called a concatenated key (McFadden & Hoffer, 1991:113-114). A key combination of Borwr_no and Ddate, for instance, produces a concatenated key, ordering due_date within the borrower.

Date (1983a:182) mentions that the objects in a relational model are relations and domains which correspond to entities and attributes but does not contain any interrelational links or access paths as they are not the basic object types defined for the Relational model. The following characteristics are defined for relations:

- Each column contains values about the same attribute, each cell value (intersection of column and row) must be simple ie. containing only one value, for example, 123 and not multiple values, for example, (123,234,456). Each attribute is atomic.

- Each column has a distinct name (attribute name) within the relation, otherwise it must be qualified by the relation name, for example, customer.cust_no but cust_no
is distinct within the relation with no two columns having the same name. The order of the columns is immaterial.

- Each row is distinct, that is, all the attributes taken together identify a specific entity, although the key values may be the same for some entities, the combined attributes yield a unique entity. The order of the rows is immaterial (McFadden & Hoffer, 1991:113).

The following relations are defined for the Issues system:

BIB (*Brn, Title, Author, Class, Note)
ITEM (*Acc_no, +Brn, Price, +Borwr_no, Ddate, Od)
BORROWER (*Borwr_no, Name, Tel, Add, Status)

In the Issues example, the Bib, Item and Borrower are basic relations that exist independently of all other data, their primary keys are denoted by attributes prefixed by an '*' while foreign keys are denoted by attributes prefixed by a '+'. The Bib relation includes the Brn, Title, Author, Class and Note for a specific book but excludes the copy (Item) details. Similarly, the
Item and Borrower relations are basic relations identified by the Acc_no and Borwr_no respectively. Even though links are not basic to the model, the '+' is introduced to show linkage. The +Brn serves as a link from the Item to the Bib relation while the +Borwr_no serves as a link from the Item to the Borrower relation. Using the keys, the one/many relationships between the Bib and Item and Borrower and Item are preserved.

2.4.1.2 Data Integrity

In a relational model the links are not 'hard-wired' as in the case of the hierarchical and network models where the links define the relationships as part of the model. Pointers and access paths are used to navigate through the database in the network and hierarchical models. The relational model uses common keys from different tables to link and match select attributes common to two or more files. However, the defined common attributes must have a common domain to properly relate the tuples, the names of the attributes can, however be different (Mcfadden & Hoffer, 1991:115-6). Links and relationships are established during the search, making the relationships implicit and not explicit as in the network or hierarchical models (Hunter, 1985:65). Due to the lack of explicit links more stringent integrity checks are
required in the relational model. The integrity part of the model consists of two general integrity rules that are applicable to all databases, defined as follows:

1. Every table should have a primary key - that is, a field, or field combination, that serves as a unique identifier for the records in the table. Primary keys should not accept null values.

2. If table T2 includes a foreign key FK matching the primary key PK of Table T1, then every value of FK in T2 must either (a) be equal to the value of PK in some record of T1 or (b) be null (Date, 1983b:251)

In the example on issues, every Borwr_no in the Item relation must have a Borwr_no in the Borrower relation. The integrity is violated when no PK exists for a given FK.

2.4.1.3 Manipulation

The manipulative part of the relational model consists of a set of operators known collectively as relational algebra, (Date, 1986:27) of which the five basic operations are, the union, selection, difference, cartesian product and projection (Ullman, 1982:152-153).
Each operator performs data manipulation onto one or two tables at a time, producing a new relation as a result. When queries are complex they must be broken down into simpler commands, as a series of binary or unary algebra commands (McFadden & Hoffer, 1991:571). For each of the operations except the Cartesian product the number of attributes must be the same and defined on the same domain (union-compatible) (Date, 1986:27).

Consider the Relations A and B as shown in Fig 2.5. Each relation has three attributes.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>a</td>
<td>f</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>d</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>g</td>
<td>a</td>
</tr>
<tr>
<td>d</td>
<td>a</td>
<td>f</td>
</tr>
</tbody>
</table>

**Fig 2.5 Relation A**  
(Ullman, 1982:153)

**Fig 2.5 Relation B**

The union of two (union-compatible) relations A and B is the set of tuples T that belong to either A or B or both (See Fig 2. 6a). The difference between two (union-compatible) relations A and B is the set of all tuples belonging to A and not B (Fig 2.6b). The product (Cartesian Product) of two relations A and B is the set of all tuples T such that T is the concatenation of a tuple a belonging to A and a tuple b belonging to B (Fig 2.6c). The projection is a subset of columns in a given
relation. The subset is obtained by selecting specified attributes and then eliminating duplicate tuples (Fig 2.6d illustrates a projection on A). A selection operator yields a set of tuples T, that is a subset of tuples of a given relation (Fig 2.6e). The table oriented approach of the relational model, enables tables to be combined in complex ways, using the file manipulation operators of select, union, projection, difference and cartesian product.

2.6a Union

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>f</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

2.6b Difference

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>a</td>
<td>f</td>
</tr>
<tr>
<td>d</td>
<td>a</td>
<td>f</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>d</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>d</td>
</tr>
</tbody>
</table>

2.6c Product

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>a</td>
<td>f</td>
</tr>
<tr>
<td>d</td>
<td>a</td>
<td>f</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>d</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>d</td>
</tr>
</tbody>
</table>

Relational models are simple models, represented as flat tables but the real-world does not necessarily present itself in nice neat tables (Brooks, 1985:5). The real-world environment has to be fitted into relations.
One of the drawbacks of relational databases is precisely the strict need to normalise data to eliminate repeating groups, and creating new files for them\textsuperscript{11}. Unlike systems such as PICK, multi-valued items within a field in the form of a group item, are allowed, for example, all data on an author, his surname, initials, dates of birth, death etc. are stored in one field but are separated by subfield identifiers\textsuperscript{12}.

The lack of implicit relationships in the model, makes the model more flexible to answer a wider variety of requests not anticipated before. However, such flexibility can lead to meaningless queries and long search times when relationships are not well defined. Further, the use of foreign keys increases the danger of loss of referential integrity. According to McFadden & Hoffer (1991:118) relational tables have been found to be slow especially for transaction processing but efficient for less structured query processing.

Unlike the E-R model and other database models, there is a lack of semantic quality control in the relational model. This makes it difficult to interpret the tables in it’s table-driven form. In the E-R model for instance, relationships are defined and expressed in terms of
diamonds relating one or more entities. In the network model relationships are defined using a set that forms a link between the owner (one end of the relationship) and members (many end of the relationship). Further, additional rules can be charted on the model, for example, a member may be either mandatory or optional or even mandatory automatic or manual.  

2.5 Dbase Database Management System

Dbase is the most widely used database management system in the world today. It’s ease of use and dependable performance have contributed to it’s success (Hayman, 1990.ix). As a DBMS, Dbase caters for a wide variety of tasks, providing a wealth of commands and functions. Powerful facilities such as global updates; standard language constructs similar to those provided in the Fortran and Basic languages; different modes of interfacing the system, namely, the assistant, dot prompt and programming interfaces, are advantages. The ease of defining, setting up and using data files, indexes, screens, report forms and querying facilities makes the software user-friendly and easy to use. The question of whether to use Dbase III+ to handle a specific task, is however, not so simply answered (Beiser, 1987:171).
In order to evaluate Dbase as being adequate for a given application, it’s capabilities need to be understood first. The subsystems discussed by McFadden & Hoffer (1991:360-362) provides a guideline and point of departure for the evaluation of Dbase as a DBMS. For the purposes of the study, the subsystems are regarded as utilities and functions as they are not fully fledged subsystems\textsuperscript{14}. While an evaluation of Dbase as a DBMS is an important aspect of the study, all facets cannot be covered adequately. Instead key aspects, namely, the Dictionary, Screen Painter, Report Writer, user views and indexing would receive attention. These key aspects, once integrated, forms the basis of any application, enabling data to be defined, input, updated, deleted and retrieved. Dbase III+ does not have a performance, security, integrity or application development subsystem (Dbase IV does) and a discussion of these aspects are therefore omitted.

2.5.1 Primary Interfaces

There are three primary ways of interrogating and using the Dbase system. They are the Assistant, the Dot Prompt and the Programming language interface, of which the Dot Prompt is the most important serving as a common link to the other modes\textsuperscript{15}. The assistant provides users with
easy-to-use pull-down menus that enable the novice with no exposure to Dbase or a programming language to execute complex commands through the selection of a series of simple commands from pop-up menus. With the Assistant, the user can open databases and indexes, set up query views, format and screen files, modify the database and generate reports from them.

For those who find the Assistant too slow and a more flexible command interface is required, use of the Dot Prompt is recommended. The user is able to, for instance, run applications using the Do <Program-Name> command and issue DOS commands. All facilities available at the Assistant are available at the Dot Prompt. Even though the Dot Prompt has much more facilities than the Assistant, it does not allow greater control of the order of the processing. Aspects like loops, conditional statements and procedure files that control the order of processing are available under the programming language interface. The Dot Prompt is limited by the length of the command line and the number and complexity of the commands that may be issued. Just as commands were entered, at the Dot Prompt to execute a task, the commands are entered into a command file using the Dbase built-in editor or any word processor. The full range of
facilities available at the Dot Prompt are available in the programming environment. The Dbase programming environment forms the basis of discussion for the remainder of the chapter. Instead of discussing the programming language directly, each element that forms part of an application are introduced and discussed. These elements are then combined into an integrated application.

2.5.2 Dictionary
The Dbase dictionary is integrated with the data and forms part of the header of a file. When a file is open the definitions are known to the DBMS. The header contains specific control information such as the number of records in the file, the date the file was last updated, the total record length, the name, type and length of each field. The header is variable in length and is able to accommodate a varying number of fields. Following the header, is the data for each record, which are of fixed length. Records do not have an end of field or record marker. Instead records are stacked one after the other in record number order. Each record can be computed by the system, for example, a header of 120 characters and record length of 100 characters (including the deletion marker), imply that the first record starts
at position 121 (immediately after the header and includes the next 100 characters); the second at position 221 (120+100+1) etc.

The Dbase dictionary utility is able to create, update and delete record, index and view definitions. Besides these functions, the dictionary facility can display the structure of the database; modify the structure, and copy file structures. These functions are however, not part of one utility but forms part of a wider 'utility-based' system, for example, files (and hence record and index definitions) can be deleted using the standard file deletion command 'Delete <Filename>' which is used to delete any file including database files (*.Dbf), memo files (*.Dbt) index files (*.ndx) and program files (*.Prg)\textsuperscript{16}.

Using the Create table <Filename> command at the Dot Prompt or by selecting the create table command using the Assistant, a fill in the blanks form is presented to the user to fill in the corresponding field name, its type (character, date, numeric, logical or memo) and the length of each field (maximum of 254 characters for fixed length fields). The utility does not allow duplicate
field names. The following are the file definitions for the Borrower, Item and Bib files of the issues system (Fig 2.7):

<table>
<thead>
<tr>
<th>Field</th>
<th>Field Name</th>
<th>Type</th>
<th>Length</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BRN</td>
<td>Numeric</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>TITLE</td>
<td>Character</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>AUTHOR</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CLASS</td>
<td>Character</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>NOTE</td>
<td>Memo</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

** Total ** 62

Fig 2.7(a) Bib File Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Field Name</th>
<th>Type</th>
<th>Length</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ACC_NO</td>
<td>Numeric</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PRICE</td>
<td>Numeric</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>DDATE</td>
<td>Date</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>BRN</td>
<td>Numeric</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>BORWR_NO</td>
<td>Numeric</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>OD</td>
<td>Logical</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

** Total ** 25

Fig 2.7(b) Item File Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Field Name</th>
<th>Type</th>
<th>Length</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BORWR_NO</td>
<td>Numeric</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NAME</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADD</td>
<td>Character</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>TEL</td>
<td>Numeric</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>STATUS</td>
<td>Numeric</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

** Total ** 47

Fig 2.7(c) Borrower File
The record definition accommodates fields that are numeric, character, date, memo or logical. Of particular interest to library applications, is the memo field which is helpful for recording comments, annotations, notes, abstracts or any unstructured data. To simplify space management such fields are not stored with the other fields but in another file (file with dbt extension). A pointer of 10 characters is maintained to link the memo fields to the records in the dbf file.

2.5.3 Index Definition
Indexing allows data to be ordered and displayed in more than one sort order without the need to sort the database itself. Indexes are defined on one or more fields or columns. To define an index, a Dbase file (.dbf) must be active (that is, in use). The index is automatically created and maintained by the system. To activate an index, the command 'Use <Filename> Index A, B, C' associates the file with the indexes A, B and C. All updates to the files cause changes to the indexes whenever the data on which the indexes are defined, are changed. When data is changed and the associated indexes are not opened, the indexes would require reindexing. Indexes may be alphabetic, numeric or chronological.
Index entries on only the first instance of a duplicate entry are allowed, using the unique attribute (see example below). The general syntax for index creation is 'Index on <Key expression> to <file-name> [expression]'.

The following is an example:

. Use Bib
. Index on Brn to Bib

In the above example, the Bib data file is opened (line 1) and an associated index, Bib.ndx (line 2) is created on the field Brn. Once the index is created, the command 'Use Bib index Bib' is used to permanently associated the data file with the index. The examples below illustrates variations in indexing techniques. In example one below, the date is converted to a string expression and concatenated with another string, the Borwr_no, to list items by Borrower within date, in ascending order. In example 2, the title is indexed on upper case (upper).

Indexing is sensitive to case and for searches to be successful, both lower and upper cases must be in one standard format. This is usually effected by adhering to one uniform case (upper or lower). Duplicate entries are excluded from the index entries in example 2; the 'Unique' attribute restricts indexing to only the first
instance of a duplicate key. The attribute is useful in library applications to validate entries, for example, checking whether an incoming title is a duplicate or not.

1. Index on Dtoc(Ddate) + Str(Borwr_no)
2. Index on Upper (Title) Unique

Dbase index files are organised as B+ tree data structures with each entry in a Dbase index containing an attribute value and the address of a single record with that value (Pollard, 1988:59). Each B-tree adopts a method of creating a hierarchy of indexes, in the form of an inverted tree with multi-levels (Ullman, 1982:58-59). A B+ tree organisation is efficient both in terms of storage, retrieval and speed of access. All the index entries for a particular attribute value are 'grouped together' in one logical area of the index. When a request is made to search a tree for a given key, the search begins from the centre, splitting the index into half which compares the key and selects the relevant half, which is again split into half until the desired match is found or the search is exhausted. Such an approach leaves no margin for error and a matching entry can be found within a few searches. Once the first matching entry is found for a particular search key, each
subsequent match is found by sequentially stepping through the index until a non-matching entry is found (Pollard, 1988:59).

Comer indicates that a Btree of order fifty that indexes a file of one million records can be searched with only four disk accesses in the worst case, and it requires at most one disk access to locate the next key in indexed order (Pollard, 1988:60). Despite the speed and compactness of the Dbase indexes, Dbase files do not support indexing on multi-valued items, for example, a field with the following values per field:

HYDROLOGY ; GEOLOGY ; HISTORY
GEOLOGY ; HISTORY ; CHEMISTRY

In a single-file mapping, as illustrated above, an index on the first field will retrieve Geology and not History because substring or partial key searches work only if the search expression matches the index key expression starting with the left most character (Pollard, 1988:57). To retrieve History a sequential search has to be made on each record to find a match. To effect retrieval, each of the multi-valued data items must be used as keys in an index.
2.5.4 Views and Relationships

'A view is a virtual, or imaginary table' (McFadden & Hoffer, 1991:66). A table is composed of one or more related columns taken from one or more tables. A view, therefore, is only part of a total system. The following is a simple view:

```
. Use Borrower
. List Borwr_No, Name, Add
   for Name = 'Smith'
```

The view provides a list of borrowers from the Borrower file for names starting with 'Smith'. In using views, some qualification or filter condition may be specified to restrict the rows to be included in a view. In the above example, only records starting with 'Smith' are extracted. The attribute 'Tel' is omitted from the view definition.

A view or virtual table acts as a window through which the user can see only the items listed. Irrelevant data is concealed from the end-user. Further, a view insulates the user from any changes in the logical structure of the database, for example, if a field F is moved from table T1 to table T2, the new view redefines the view giving the user the impression that the files are not altered.

67
and that F is still in T1. The new view definition conceals the physical structures from the user (Date, 1986:14-15).

Tables do not exist on their own but participate with other tables to form relationships, for example, a relationship linking the Accession Numbers per bibliographic record using the Brn as a linking field. Relationships in a view are set up using the 'set relation to' command. A more complex view for the issues subsystem is defined as follows:

1. Select 3
2. Use Borrower index Borrower
3. Select 2
4. Use Item index Item
5. Set relation to borwr_no into Borrower
6. Select 1
7. Use Bib Index bib
8. Select 2
9. Set relation to Brn into Bib

In the above view the Item in workarea 2, is the master control file. The remaining files are denoted by work areas 1 (Bib file) and 3 (Borrower File) respectively. All records and relationships are determined relative to the Item records, that is, each time a record in the Item file becomes the new current record, the record in Bib file moves to a record having the same Brn (common key). If there is no match the pointer is positioned at the end.
of the Bib file. At the same time, the record pointer in
the Borrower file is moved in concert with movement in
the Item file but based on the content of the Borwr_no
field (Beiser, 1987:13).

The first 4 commands opens the Borrower and Item files
with associated indexes. The relationship between the
Item and Borrower files is defined using the 'Set
relation to Borwr_no into Borrower'. Both files are
linked by a common Borwr_no. The relationship is set from
the master control file (work area 2) to the Borrower
file. Similarly, by opening the record and index files
for the Bib record (lines 6-7), workarea 2 is again
selected as the control file (line 8), a relationship is
set up with the Bib file using the Brn as common field
(line 9). Now each item is linked to one unique Bib
record and one and only one Borrower.

However, Dbase cannot handle complex views as illustrated
above. Defining a view does not guarantee that reports
produced would yield the desired results. For instance,
by setting the field list to Bib->Brn, Title,
Item->Acc_no, Borrower->Borwr_no, Borrower->Name and
Borrower->Tel, a simple List command to extract a report
for all records with Brn less than 15 using the filter
condition 'Set filter to Brn < 15' cannot be produced correctly due to the manner in which the files are linked. Consider the following example taken from Beiser (1987,13):

- Select 3
- Use Register Index Reglast
- Select 2
- Use Reserves Index Rsvbibno
- Set Relation to Borrowerno into Register
- Select 1
- Use Bib index Bibttl
- Set Relation to Bibno into Reserves

Beiser comments that, each time a record in bib.dbf becomes the new current record, the pointer in Reserves.dbf moves to a record having the same Bibno and at the same time, the record pointer in the Register.dbf is moved in concert with moves in the Reserves.dbf but based on the contents of the field Borrowerno (Beiser, 1987:13). This kind of motion is made possible because the Register records are dependent on the Reserves which are in turn dependent on the Bib records and that the Bib.dbf is the primary control file in the view.

In the view defined above the Item file is the control file. Selecting work area 1 or 3 would yield incorrect results. For example to use the Borrower file (Workarea 3) as the control file, replace 'Item' by 'Borrower' in the relation 'Set relation to Borwr_no into Borrower' and
add the command 'Select 3' to the view (last line). Now the Borrower points to the Item which in turn points to the Bib records. However, such a fixed view when using a report generated by Dbase would make only one pass through the Borrower file and finds an associated record in the Item and Bib files. However, only the first Item is selected and reported on, omitting reports on more than one item per borrower, which is erroneous. The one-to-many relationship between the Borrower and Item (see discussion of relationships in chapter 3 section 3.4.2.2 of this thesis). Greater control is required on the order of processing and the way reports are produced. This issue is considered under section 2.7 of this chapter. Fig 2.8 illustrates a simple report using the view below for the fields Brn, Title, Author, Acc_no and Borwr_no for Acc_no < 10, using the Item as the control file:

Select 1
Use Bib Index Bib
Select 2
Use Item index Item
Set relation to Brn into Bib
List Bibliographic details for Accession Number < 10

<table>
<thead>
<tr>
<th>BRN</th>
<th>TITLE</th>
<th>AUTHOR</th>
<th>ACCESSION</th>
<th>BORROWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Practical systems design</td>
<td>Daniels, A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>Practical systems design</td>
<td>Daniels, A</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>20</td>
<td>Practical systems design</td>
<td>Daniels, A</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Essential guide to Dbase III+</td>
<td>Beiser, Karl</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>Data base</td>
<td>Atre, S</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Essential guide to Dbase III+</td>
<td>Beiser, Karl</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>25</td>
<td>Computers in Business Managment</td>
<td>O'Brien, J A</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>25</td>
<td>Computers in Business Managment</td>
<td>O'Brien, J A</td>
<td>9</td>
<td>5</td>
</tr>
</tbody>
</table>

Fig 2.8 Field List from Bib View Using Report Form Bib

The creation, update and deletion of views, indexes and records makes the Dbase dictionary utilities easy to use. However, the dictionary suffers from a few drawbacks, for example, rules or constraints cannot be defined as part of the dictionary which is the highest level of control for data. A range of values is instead stipulated at a lower level of control, namely, the screen interface (or programming level) where each template picture clause controls the type of data and acceptable range of values.
While a dictionary is provided, it cannot be considered a subsystem, as it exists at a very rudimentary level. Ideally, an element in the dictionary should be defined only once, for example, field names, data lengths and types for the title, author and classification number are recorded only once, irrespective of the files and records that use the elements. Dbase, on the other hand, defines each element within each record structure, the elements do not exist independent of the record structure. Hence, it is possible to define the length of a title in one file having 200 characters and 150 characters in another. The failure of a dictionary to define a common pool of data elements to serve as building blocks to database definition imply the possibility of inconsistency in data element definitions.

2.5.5 Query and Data Manipulation

Data manipulation or retrieval is the process of constructing tables from one or more tables using the relational operators defined earlier. Dbase is not fully relational\textsuperscript{17} that is, not all of the fundamental operators, Select, Project, Union, Difference and Product, are present. The 'Product' operator for instance, is excluded in Dbase. The Dbase commands Find, Seek, Locate and Print are used to search and manipulate
one or more tables. To facilitate the search, indexes are used to order the tables and link them. Each of the operators are considered in turn.

**Select**

In a select operation some or all or none of the rows of one or more tables are produced as a result of a condition being specified on one or more attributes. Fig 2.9a illustrates the Select command for a list of Borrowers for Borwr_no > 5.

**Project**

The project operator lists one or more columns from a table, for example, the command ‘List Borwr_No, Name, Tel’ lists selective columns from the Borrower file (Fig 2.9b).

<table>
<thead>
<tr>
<th>Borwr_no</th>
<th>Name</th>
<th>Address</th>
<th>Tel</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Leppin, GW</td>
<td>491 Sydney Rd Umbilo</td>
<td>252147</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Philip, CN</td>
<td>85 Buckingham Rd KL</td>
<td>7644672</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>Chetty, K</td>
<td>23 Firozepre Rd Mbknk</td>
<td>421544</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>Philemon, S</td>
<td>70 Detroit St Hovnsde</td>
<td>4006332</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Phillips, CL</td>
<td>152 Edgeley Rd DbnN</td>
<td>835676</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>Bilchick, Simon</td>
<td>35 San Remo</td>
<td>324626</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>Bidder, A</td>
<td>45 Haslam Rd Qnsggh</td>
<td>4640292</td>
<td>1</td>
</tr>
</tbody>
</table>

**Fig 2.9a Select Operation**
. List BORWR_NO, NAME, TEL
RECORD  BORWR_NO  NAME          TEL
1        1       Lennard, ND    473084
2        7       Leppin, GW    252147
3        5       Chetty, K     4004274
4        6       Philip, CN    7644672
5        8       Chetty, K     421544
6        10      Philemon, S   4006332
7        11      Phillips, CL   835676
8        13      Bilchick, Simon 324626
9        15      Bidder, A     4640292

Fig 2.9b Project Operation

Join Operator
The join operator concatenates tuples from two or more relations based on a common matching column defined on the same domain. A join requires the use of an index to link the files and one or more 'set relation to' commands are issued. The view defines the joins. By using the List command, selected rows can be extracted. The view defined in the previous section illustrates a join (Fig 2.8).

Union
Two tables that have corresponding columns with the same domains can be merged into one table to form a Union. Duplicate tuples are removed in the process. In Dbase III+ and Dbase IV, the command closest to the Union operator is the 'Append From' command. The command copies rows from one table to the end of the active table.
Duplicate rows are not removed. If the 'With' command is specified only field names found in both tables are used and therefore Unions require the same name and type when a qualification is used. The 'Append From' command is useful to import data from non-dbase files.

**Difference Operator**

Dbase does not have a difference operator - the operator is used to find the difference of two tables A an B to yield a third table C, which contains tuples that are in A but not in B. Such a operator can be used to answer queries like "Which patients have been treated by the physician 'Smith' and not by the physician 'Jack'?"

In Dbase the typical insert, delete and update operations are easily performed by the Union and Difference operators. An 'Append Blank' command performs a Union of a tuple with the current file by adding a blank record at the end of the file. Thereafter the field details can be replaced with appropriate fields\(^{19}\). Similarly, a difference is effected when a tuple B is deleted from an existing table A. In this instance, table B consists of one tuple. A deletion marker is placed in the record of table A which logically deletes the tuple. The record is only physically deleted when the file is packed.
Unions and differences as defined by the relational algebra, are not satisfactory for database insert, update and delete operations because they do not handle error situations appropriately, for example, an attempt to add or delete an attribute when it is not defined for a given tuple. Hence systems supporting the relational operators must provide explicit operators (for example, Append, Replace and Delete in Dbase) that provides appropriate messages or even reject illegal operations (Date, 1987:213).

2.5.6 Screen and Report Design

The design of reports and screens are just as important as ensuring that proper controls are instituted during insert, update, delete and retrieval operations. Users interact with screens and obtain information from reports. Hence the screens must be user-friendly and the reports accurate. Both formats must be acceptable to the end-user. The Dbase DBMS has a screen painter and a report generator to facilitate screen interaction and report generation.

With the screen painter, a facility to customise data entry is provided. When using the screen painter, fields can be moved anywhere on the screen, with messages,
instructions and comments displayed on the screen. Additionally, graphic designs such as single or double-line boxes can be used to enhance display (Chou, 1986:84-85). The resulting data entry form is then saved as a format file (with extension fmt). The format file is used to append and update the database records after duly opening the appropriate database and index files.

Data entry ranges from a simple ‘fill in the blanks’ utilising appropriate prompts to accept data (Chun, 1987:43) to a more sophisticated command language interface using menus, language constructs and format files. Using the ‘?’ command to display data is cumbersome as the data is placed immediately below the command line and not at a desired position on the screen. The Dbase ‘@... say’ commands, on the other hand, displays data at a specific position on the screen using the following general format ‘@<row>,<column> Say <data item>’. The following are examples:

@5,10 Say Title
@7,12 Say Acc_no Picture '99999'
@12,20 Say Class Picture '!!!!!!'
@14,2 Say 'Borrower Number is : ' + Str(Borwr_no)

In the first case the title is displayed at line 5, column 10 while in the second case, only digits are displayed for the Accession number. In the third example,
the classification number is displayed in upper case. In the fourth case a numeric variable is converted to a string using the string function (Str) and is concatenated with the string 'Borrower Number is: '. Fig 2.10 illustrates a screen designed for the Borrower

![Borrower Record Entry Form]

<table>
<thead>
<tr>
<th>Number</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Chetty, K</td>
</tr>
<tr>
<td>Address</td>
<td>23 Firozepre Rd Mbnk</td>
</tr>
<tr>
<td>Telephone</td>
<td>421544</td>
</tr>
</tbody>
</table>

**Fig 2.10 Screen for Borrower Records**

The report generator operates on similar principles as the screen painter. Reports can be generated on paper as well as on the screen (retrieval). A title or header is provided for the report. Each column can be sized separately with the option of stuffing one column with one or more fields (for example, surname and initials belonging to different fields). Where the data is too long and does not fit on one line, it wraps around to the next line automatically (Cowart, 1986:119-124). The columns can be ordered using indexes or sorted records.
(with totals and subtotals calculated and displayed). In addition, one or more rows in the view can be selected by defining a selection criteria, for example, 'List Borwr_no for Status = 3'\(^2\). Fig 2.11 shows the associated Report for the Borrower file (Borrower.Frm).

---

Page No. 1
08/02/93

LIST OF BORROWERS WHO ARE STAFF MEMBERS

<table>
<thead>
<tr>
<th>STAFF NAME</th>
<th>ADDRESS</th>
<th>TELEPHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Lennard, ND</td>
<td>32 DoverRd Wntwrth</td>
<td>473084</td>
</tr>
<tr>
<td>8 Chetty, K</td>
<td>23 FirozepreRd Mbnk</td>
<td>421544</td>
</tr>
</tbody>
</table>

---

**Fig 2.11 Report Form for Borrower Records**

2.5.7 *Dbase Programming Language*

One or more simple commands are not adequate to build most information systems. Hence, the reliance on stored programs containing commands to control the sequence of operations, set up the database and screen interfaces, to add, update, delete and retrieve data in a database. Most DBMS support the stored program concept to perform repetitive tasks and enable sophisticated searches and record selection. Such systems may provide various types of interfaces, for example, simple user-friendly menus affording a selection of options in the form of pull-down
menus. In short, writing programs control and determine the sequence of operations of the various processing requirements. The program puts together the various objects into one integrated whole. Each set of commands are processed in sequence (top-down) until special language instructions (for example, If... Then...Else) interrupts the normal sequence of instructions. It is not possible to demonstrate all the features of Dbase programming. Instead only key aspects are explained and demonstrated.

**Procedures**

To invoke a Dbase program or procedure, the command 'DO <program-name> with <parameter list>' is used. One program may be embedded within a larger calling program, with parameters being passed as an option. All variables created within the calling program are available to the called program and any program it may call. Changes made to the variables by the called programs are passed back to the calling program. However, variables created within the called program are lost when the program terminates (Liskin, 1987:232). The parameters in the programs need not have the same names but must be the same in type and number of parameters.
A subroutine is a sequence of commands that can be used and called by more than one procedure, its main objective is to perform a simple task that acts as building blocks to more complex tasks. Therefore, it is common place for one procedure or program to call many subroutines, the procedure itself co-ordinating the various processes. The following subroutine, Overdue (Fig 2.12a) ascertains whether an item is overdue. The application demonstrates the interaction of the dictionary, views and reports in generating a list of overdue books. The Overdue Status (Od) and Ddate (Due-date) are passed as parameters from the calling program (Procedure Over).

The Overdue subroutine ascertains whether an item is overdue with respect to a given cut-off date (Cdate). Dbase does not have a utility to make comparisons on dates. Hence the overdue subroutine compares the cut-off date with the due-date and if the due-date is less than the cut-off date, the item is considered overdue and the status is set to true (the default setting is false). The Od parameter is passed back to the Over program (Fig 2.12b).
The Overdue subroutine is placed in a procedure file (Procs.Prg) for use by any program that may require its use. Commonly used utilities are placed in a procedure file, each procedure, accepting zero, one or more parameters. The procedure file must be declared to the program first before use. This is effected by using the 'Set Procedure to <file name>' command. The procedure itself, is invoked by the command 'DO <filename> With <Parameter list>'.

** Procedure Overdue **

** This Procedure file holds all procedures used in the application. Only one Procedure is however, defined **

** Subroutine accepts parameters Ddate1 (Due-date), Od1 (Overdue Status), Cdate (Current Date/Date of **

** Request) and adjusts the value of Od1 which **

** is then updated in the calling program **

PARAMETERS Ddate1, OD1, Cdate

** Dates, Ddate1 & Cdate) are converted to days, **

** month and year. **

Days1 = Substr(dtoc(Ddate1),4,2)
Mnth1 = Substr(Dtoc(Ddate1),1,2)
Yr1 = Substr(Dtoc(Ddate1),7,2)
Days2 = Substr(Dtoc(Cdate),4,2)
Mnth2 = Substr(Dtoc(Cdate),1,2)
Yr2 = Substr(Dtoc(Cdate),7,2)

Do while .T.
If Val(yr1) = 0 & & When book not borrowed date
Exit & & set to 0. Hence Od1 = .F.
Endif

If Val(yr1) > Val(yr2) & & If Due_year greater
Exit & & than current year program is exited
Else
If Val(yr1) < Val(yr2) & & If Due year is less than
Od1 = .T. & & current year then item is overdue
Exit
Endif

83
** At this point the Due year equals that of the
** current year. Only Tests for the month and
** days are essential
If Val (mnth1) < Val(mnth2)
  Od1 = .T.
  Exit
Else
  If Val(days1) < Val(days2) .and. Val(mnth1) =;
    Val(mnth2)
    Od1 = .T.
    Exit
  Endif
Endif
Endif
Exit
Enddo
RETURN

Fig 2.12a Overdue Subroutine within Procedure file Procs

**Over.prg
** Aim: Calculates Overdue Status & prints Report

**Initialises environmental conditions
Set Status Off
Set Talk Off
Select 1
Use Bib Index Bib
Select 2
Use Item Index Item
Select 3
Use Borrower Index Borrower
Select 2
Set Procedure to Procs

** Accept a valid Date
Cdate = Ddate
Clear
@12,0 Say 'Enter Cut-Off Date for Overdue Books: ' Get
Cdate
Read

** Sets up Temporary Index to sort by Borrower Number
Index on Borwr_no to AccBor

Go Top
Do While .not. EOF()
  ** Loop updates Overdue field (OD) in database
  ** before report is printed
  OD1 = .F.  && Sets overdue to false (default)
Ddatel = Ddate
Do Overdue with Ddatel, Od1, Cdate
   Replace Od with Od1
   Skip
Enddo

*** Sets up Header for Report
? ' LIST OF OVERDUE BOOKS'
?
?
? ' BORROWER NAME      TITLE     ACCESSION NUMBER
? ' NUMBER             DUE-DATE
?

Select 2
Go Top
Do while .Not. EOF()

If Od && If Od is true, the date and Accession
Ddatex = Ddate && number accepted from Item Record
Acc_nox = Acc_no

** Finding Name of Borrower
Namex = Space(15)
Borwr_nox = 0
If Borwr_no > 0
   Borwr_nox = Borwr_No && Borrower Number accepted
Select 3
Seek Borwr_nox
If Found()
   Namex = Name && Borrower Name accepted from Borrower
Endif
Endif

** Finding Title of Book Borrowed
Select 2
Brnx = Brn
Titlex = Space(10)

If Brn > 0
   Select 1
   Seek Brnx
   If Found()
      Titlex = Title && Title accepted from Bib record
   Endif
Endif

** Now Printing Line if Od = .T.
   If Borwr_Nox > 0

85
? Borwr_nox, Namex, Titlex, Acc_nox, Ddatex
   Endif
Endif
   Select 2
   Skip
Enddo
Close All
!del Accbor.ndx  & Deletes temporary Index
RETURN

**Fig 2.12b Over Program using Overdue Subroutine**

The Over program is the control program consisting an initialisation segment followed by the program proper. As part of the initialising process, the environmental conditions Set talk Off, Set Status Off, are set. The procedure file is assigned using the command 'Set Procedure to Procs'. All files and indexes to be used by the application are opened and declared. The Bib, Item and Borrower files have been created from the Assistant. The data for each file has been captured using the Browse and edit facilities.

The main segment of the program following the initialisation, is a request for a valid Cdate. Thereafter, a Do While loop controls the update of the values of Od. Depending on the overdue status, the field is changed accordingly in the Item record, replacing the old Od value for the new one. The loop itself is exited after the end-of-file (Eof) is encountered when control
is transferred to the first statement following the
Enddo, otherwise the loop is re-entered and a new record
is processed. Thereafter, a temporary index (Accbor) is
created to sort all records by borrower number.
Subsequently, a report is generated for all Borrowers
with overdue books. Each item with Od = .T. is listed.
Fields included in the report (Fig 2.13) are, the
Borwr_no, Name, Title, Acc_no and Ddate. Finally, the
temporary index is deleted before the program is exited.

<table>
<thead>
<tr>
<th>BORROWER NUMBER</th>
<th>NAME</th>
<th>TITLE</th>
<th>ACCESSION NUMBER</th>
<th>DUE-DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lennard, ND</td>
<td>Practical systems design</td>
<td>1</td>
<td>04/12/93</td>
</tr>
<tr>
<td>5</td>
<td>Chetty, K</td>
<td>Computers for libraries</td>
<td>14</td>
<td>07/10/93</td>
</tr>
<tr>
<td>5</td>
<td>Chetty, K</td>
<td>Computers in Business Managmnt</td>
<td>16</td>
<td>06/12/93</td>
</tr>
<tr>
<td>6</td>
<td>Philip, CN</td>
<td>Practical systems design</td>
<td>12</td>
<td>08/16/93</td>
</tr>
<tr>
<td>6</td>
<td>Philip, CN</td>
<td>An intro to database systems</td>
<td>12</td>
<td>08/11/93</td>
</tr>
<tr>
<td>8</td>
<td>Chetty, K</td>
<td>Essential guide to Dbase III*</td>
<td>4</td>
<td>06/03/93</td>
</tr>
<tr>
<td>8</td>
<td>Chetty, K</td>
<td>Computers in Business Managmnt</td>
<td>8</td>
<td>01/10/92</td>
</tr>
<tr>
<td>15</td>
<td>Bidder, A</td>
<td>Practical systems design</td>
<td>3</td>
<td>05/24/93</td>
</tr>
</tbody>
</table>

**Fig 2.13 Report: Overdue Books for Cdate = 08/20/93**

**Functions**

Dbase has a range of functions to perform different
tasks. A function transforms the input data to yield a
specific type of output. The `Ltrim()` function, trims a
string removing trailing blanks to the left of the
string; the `Dtoc (Ddate)` translates the Ddate to
characters, Substr(Title,13,11) extracts 11 characters starting from position 13 in the title string, for example, for the title 'Library and Information Science' the function yields the string 'Information'. The Upper(Substr(Title,13,11)) transforms 'Information' into 'INFORMATION' (upper case).

The use of high level functions, procedures and commands can achieve results without the user knowing how to perform a task, for example, the sort command sorts a file without the user knowing exactly how the sort is done nor does the user need to understand the sorting process (Chun, 1987:41). Similarly, the Substr() function, for instance, is used to extract a string without knowing how the function works. Hence, Dbase provides easy to use functions and procedures that serve as building blocks to more complex tasks.

**Logical Operators**

Logical operators enable the comparison of numbers and strings and determines whether a given condition is true or false. The operators < (less than); > (greater than); = (equal to); <> (not equal to) <= (less than or equal to); >= (greater than or equal to) (Chun, 1987:42) are some of the key operators defined for Dbase. These
operators allow the narrowing or broadening of any search, through selection on one or more fields. For instance, a list of undergraduates can be produced (using the example on issues) by testing whether the Status is equal to 1. The general syntax is:

Do while .not. Eof()
If Status = 1
**statements to list Borwr_no, Name, Add
Else
Skip
Endif
Enddo

The processing continues if a record with status equals 1 is found, otherwise the record is omitted and the next record is inspected for a matching condition.

**Modification Statements**

The Dbase modification statements include the commands Append, Replace, Update, Browse, Delete and Recall. The relational algebra data modification commands permit only one file to be updated at any time even though relationships with other tables may have been established (McFadden & Hoffer, 1991:583). Dbase modification commands accommodate more than one way to add, update and delete data, for example, an edit command allows one record to be modified, while a browse command allows more than one record to be modified using a spreadsheet like
format, in which specified fields can be selected for modification. The Borrower records for instance, are edited and appended using the following basic program statements.

Do while .T.
Use Borrower Index Borrower
Edit
Enddo

Do while .T.
Use Borrower Index Borrower
Append
Enddo

2.5.8 Drawbacks of Dbase III+
The variety of features discussed above, for example, separate utilities to define record structures, views, screens, reports and a sophisticated programming language are decided advantages of the language. However Dbase has a few limitations. The following are some:

- It does not support subfields, a structured device for accommodating multi-valued data items (Pollard, 1988:57);
- It is not equipped to handle variable length fields and is limited to 255 characters;
- It lacks the facility to declare arrays (This has been accommodated in Dbase IV, however);
- It is limited to only 15 open files;
- Memo fields are not searchable;
- It is not a compiler, therefore program execution is slower. 'A compiler is a piece of software that takes the original program statements and converts them to an independent, stand-alone, machine language program' (Beiser, 1987:23).

Despite these drawbacks, Dbase is the most widely used database management system in the world today. Its ease of use and dependable performance have contributed to its success (Hayman, 1990:i). It has earned the respect of many users throughout the world. Many enhancements have been made to Dbase III+ in Dbase IV which itself continue to have the limitations mentioned in points 1, 2 and 6 above.

With the latest version of Dbase (Dbase IV) some notable enhancements are evident. These include the facility to declare arrays. Memo fields are searchable. Dbase IV allows 255 instead of a maximum of 128 fields per record as in Dbase III+. The number of memory variables is increased from 256 in Dbase III+ to 15,000 in Dbase IV. It has a storage capacity of 64k for memo fields instead of 4k as in Dbase III+. The command line lengths has been increased to 1024 characters, instead of 254 characters
as in Dbase III+. Dbase IV has enhanced features like SQL for interrogating the database and performing individual and global updates. Furthermore, it has an application generator to generate applications from objects already defined. Despite these enhancements, Dbase IV lacks sufficient speed of processing even though the programs are 'compiled' and fails to accommodate variables above 1k characters. Due to many bugs affecting screen displays or failing to deliver promised functions, it has deterred users from implementing Dbase IV (Hayman, 1990:xv).

Many drawbacks inherent in the language are overcome by extensions provided by the Dbase clones and compilers like Database, Foxbase (Compiler run on Unix), DBXL, Clipper (language & compiler), Quicksilver (compiler) (Hayman, 1990:xvi). Many of the Dbase clone languages are a superset of the Dbase language, and include many features that cannot be handled by Dbase (McFadden & Hoffer, 1991:647).

Clipper, for instance, is a stand alone language, with facilities similar to Dbase. The same commands (with slight variations) that are used in Dbase III+ are also used in Clipper. Hence, programs written in Dbase can be compiled in Clipper. The compiler checks the program
for illegal or ambiguous statements and produces a set of object files as a result. The object files are then linked together to form an executable file, which can then be run directly from DOS (Disc Operating System), avoiding the need to use the Dbase software while the application is run. In this way, independent applications can be developed and made available to users to run under the DOS environment.

Few immediate concerns are expressed concerning bibliographic records. Bibliographic records are usually of variable length, for example, large notes. A full NAREM record averages 1.5k. Firstly, Dbase III+ and Dbase IV can accommodate at most 255 and 1024 characters respectively for variables. This makes concatenating variables and fields above those lengths impossible. Clipper, on the other hand, can handle up to 32k characters per variable. Many of the features available in Dbase IV are also available in Clipper. In addition, Clipper has the facility to declare arrays and memo fields are searchable. Up to 2048 memory variables can be handled in an application. A Clipper data file can have as many fields as allowed by RAM. The file extensions are not limited to 'Dbf' but any extension can be used provided the extension is supplied each time a file is
used. Clipper supports up to 254 work areas with ninety nine open data files. Up to 15 index files can be accommodated in each workarea. Clipper uses, in addition, index files (with extension ntx), for extra speed. Procedures can be placed anywhere, even in the same program file they are called from and is not limited to 32 procedures which must be placed in a procedure file, as specified by Dbase. Clipper programmers can define their own functions and declare the libraries as part of the program at start-up for frequently used routines and functions (Tiley, 1988:16-21). Hence when considering any enhancements, preference is given to Clipper than to Dbase IV.

NOTES
1. A master record approach would create separate records for the cataloguing, acquisitions, and serials application, ignoring the fact that common data, for example, titles, publisher details, classification numbers etc. are shared by all applications. The approach is expanded on in section 2.2 of this chapter.
2. McFadden & Hoffer (1991:21) and Cardenas (1985:14) both agree that some degree of redundancy is necessary to perform validation checks and for
performance reasons.

3. For a discussion of the network and hierarchical models consult Atre (1980:95-122) and Ullman (1982:94-144). The relational model is discussed in section 2.4.1 of this thesis.

4. An entity is defined as any identifiable object. An attribute is the characteristics of an entity, for example, the author, title, publication details, collation are attributes of the entity, book. Formal definitions are given under the Entity-Relationship model in chapter 3 of this thesis.

5. Strictly speaking, relations do not allow duplicate keys.

6. Dbase accepts duplicate keys.

7. In Dbase, the condition is reflected by the file pointer moving to the End-of-File (EOF), an error condition.


9. The examples are taken from Ullman (1982:153-154) while the definitions are adapted from Date (1986:27-30).

10. See chapter 3, section 3.5.1 of this thesis for an example of an implementation of many-to-many
relationships, for instance.

11. Normalisation is discussed in chapter 3 of this thesis.

12. In library systems, repeating groups as in the case of authors, notes, standard numbers and alternate classification numbers are common occurrences.


15. While the Assistant and the Programming language interfaces can access the Dot prompt, the Dot Prompt can access the Assistant and run applications from it.

16. Dbase files are created with dbf extensions, memo fields with dbt extensions, index files with ndx extensions and program files with prg extensions.

17. A system that supports relational databases which has a language that is less powerful than the relational algebra may be called semi-relational. According to Date (1981:214) no system is fully or semi-relational.

18. This has been elaborated in section 2.6.4 of this
thesis.

19. For examples on the use of the Replace and Append commands see section 2.5.7 of this thesis.

20. A working file with extension .Scr is created as an interim measure.

21. At the dot prompt, the report is displayed using the following command syntax: 'Report Form <name of report form file> <qualifier>', for example, 'Report form <Borwr> for Borwr_no > 5 to Print'.

22. There are a few Dbase commands that cannot be used in Clipper and can cause errors during compilation. The restrictions are discussed in the Clipper manual (Tiley, 1988).
CHAPTER 3
DATABASE DESIGN PROCESS

3.1 Introduction
Knowledge of how to design new systems, how to dissect and decompose data into logical sub-units, to know the difference between a good and bad design is desirable when a new system is being set up or when an old system is to be evaluated. Database design places emphasis on how the data itself should be analysed and managed. When designing a database, good planning prior to automation, is recommended. The present chapter explores the elements that constitute a good design and outlines the phases in the design process. This chapter identifies the various steps in the design process starting from the definition of requirements, development of an entity-relationship model and mapping this model into a relational database model. The result of this exercise culminates in the definition of files and indexes which are used to produce the desired AACR2R and NAREM reports.

3.2 Phases in Database Design
The following quotation by McFadden & Hoffer (1991:167) states concisely what is involved in database design:

'Database design is the process of developing database structures from user requirements for data. It starts
with requirements definition, which identifies user needs (present and future) for data. It then proceeds by translating these user requirements first into a logical, then a physical database design. The resulting design must satisfy user needs in terms of completeness, integrity, performance constraints, and other factors.

Database design requires the use of an organised approach to data analysis and management. Key steps in the design process include identifying and describing the data required by the users, formulating a conceptual data model (for example, an Entity-Relationship model) from the user requirements, mapping the conceptual data model into a schema or logical mapping of a particular DBMS (for example, Dbase) and mapping this again to physical storage structures such as files and tables (McFadden & Hoffer, 1991:168-170). These steps are illustrated in Fig 3.1. Each of these aspects are considered in turn.

![Diagram](image)

**Fig 3.1 Design Process**
(McFadden & Hoffer, 1991:169)
3.3 Requirements Definition

During this phase data elements are described and defined, providing details on their function, use, users, as well as the sources of the elements, for example, from customer or from the shipping department. The data is usually gathered from forms, bills, reports, existing data files and old programs which serve as a starting point for data collection (Atre, 1980:125). Information describing data such as entities, relationships and attributes are mapped graphically into diagrams such as E-R (Entity-Relationship)\(^1\) models.

During the requirements definition phase, information depicting rules or constraints under which the business enterprise operates are captured in the dictionary or directory (McFadden & Hoffer, 1991:171)\(^2\). Rules can relate to constraints such as the type of data element (character, integer), its maximum length and the range of values in the domain (for example, a domain $\geq 0$ for an integer field). In addition, data is gathered on whether the element is optional or mandatory and it's relationship with other data elements. For a business application, for instance, an invoice may only be dispatched if payment has been made. A condition of

100
payment has to be ascertained first. This can take the form of an attribute with a yes/no status to effect or deny shipment accordingly. Such decision making processes are included in the E-R diagram, where possible. Similarly, salesmen who receive a commission on sales are different from other employees who do not receive a commission. Such conditions, reflecting different kinds of employees must necessarily form part of an E-R model, the latter reflecting the reality of the enterprise. The final output of the requirements definition, are a formal set of requirements specifications that are used as input to the conceptual design.

3.4 Conceptual Design
Once the requirements are defined, the next stage is to develop a conceptual model reflecting the entities and their relationships based on the data processing needs of users in an organisation (Atre, 1980:124-5). During this phase the requirements are expressed in the form of a model, namely, an E-R diagram. The conceptual data model consolidates all the individual E-R diagrams developed during the requirements definition into one unified view.
A model is defined as 'a representation of real-world objects and events and their associations' (McFadden & Hoffer, 1991:93). Models serve as vehicles to represent data, the model itself uses a standard notation that serves as basic building blocks in the modelling process. Data models in particular are abstract representations of data about entities, events, attributes and their relationships (McFadden & Hoffer, 1991: 93). The model is usually a simplified version of the real-world, thereby facilitating an understanding of the real-world. By using a model, a structured technique is utilised to represent data.

The mere 'lumping' of data elements together is an awkward way of building files (Caswell, 1984:293). Date (1986:424-5), distinguishes between two complementary approaches to designing a database, the synthetic and analytic approaches. The synthetic approach is applied when a new database is being designed from scratch, whereby entities and their relationships are ascertained leading to a gradual build up or synthesis of the final structure. In this approach the use of the entity-relationship model is relevant. The analytic approach, on the other hand, applies when a body of data already exists in some form or other (for example, in
files) and it is necessary to analyse the data in order to understand it properly before recasting it into a more desirable form or format. The technique of normalisation provides a suitable method of constructing files. The techniques of E-R Modelling and normalisation are discussed below.

3.4.1 Data Normalisation
The master record approach identified in the previous chapter (Section 2.2) has been shown to lead to undesirable results, namely data duplication, lack of centralised control, inconsistent data, inflexible design, failure to share a common pool of data, high program maintenance and low programmer productivity. The concept of normalisation arises as a result of observing that in a dynamic database environment possible undesirable side effects referred to as anomalies due to insertion, update and deletion operations on database files, occur (Cardenas, 1985:497). The solution is to identify criteria for a good design and to establish good design practices. It is difficult to agree on what constitutes a good or the best design\(^3\). Therefore, the study focuses on elements that constitute a bad design, paying particular emphasis on ways to overcome bad design pitfalls.

103
To illustrate a bad design, consider the following example. A computerised reservation system consists of items placed on reserve by an instructor for student use. Reservations per item is limited to one per semester for a specific course with at most two reservations per semester for two different courses. Hence, provision is made for two course names and numbers. Each record in the Reserve file consists of bibliographic details (author, title); Course details (course number and course name); instructor details (instructor name and telephone number) and reservation details (semester, course and accession number). The accession number (Acc_No) uniquely identifies a particular item and hence forms the primary key for the Item record. However, for the purposes of reservations, both the semester and course details are required when placing an item on reserve. Within a semester, reservation for two courses are available, the course number alone therefore does not qualify as a key as it lacks the property of uniqueness. All these entities or objects are placed in one Reserve file with a record structure as defined below. It is assumed that one course is assigned to one instructor.

RESERVE (*Acc_No, Author, Title, *Semester, Crse_No1, Crse_Name1, Instr1_Name, Instr1_Phone, Crse_No2, Crse_Name2, Instr2_Name, Instr2_Phone)
The inherent problems of such a design becomes apparent when database operations of insertion, update and deletion are applied to the data. The instructor and course details are repeated for each item reserved and hence would occur more than once in a file. When an item is deleted, the instructor and course details are deleted. A deletion anomaly occurs when all instructor and course details are removed during the deletion process. Since the items are duplicated in different records, deletion of an item does not necessarily mean a loss of course or instructor details each time an item is deleted. However, when the last item bearing the instructor and course details are deleted and all such details are lost to the system totally, a deletion anomaly occurs.

In the above record structure, it is not possible to add course or instructor details independently without placing an item on reserve (insertion anomaly). Ideally, a file on all instructors and courses should be a permanent part of the system and not be dependent on whether an item is reserved or not. Assuming that the telephone number of the instructor is to be changed then a special program must be written to change all instances of a telephone number for a given instructor. A failure
to update all instances of the telephone number results in a update anomaly. Ideally, details on the instructor should be recorded, updated and deleted only once.

The consequences of the master record approach to file design are that data elements are bound in inflexible combinations with other data elements, new needs requiring modification of the file or the design of a completely new file (Caswell, 1984:294). For instance, the addition of one more course requires a restructuring of the file. Each of the entities, Courses, Instructors and Bibliographic items exist independent of the reservation process in the real world. Records should be designed to avoid the three kinds of anomalies caused during storage operations, that is, the inability to store a new item (Insertion Anomaly); inability to delete an item without information about an item being lost in the database (Deletion Anomaly) and inconsistencies caused by the update process, that is, when data values for the same data are different (Update Anomaly).

3.4.1.1 Definition of Normalisation
Various definitions of normalisation are proposed. According to McFadden & Hoffer (1991:218) 'Normalisation is the process of grouping attributes into
well-structured relations'. A well structured relation is one that contains the minimum amount of redundancy, allows insertion, deletion and updates without causing errors or inconsistencies arising from such operations. Caswell (1984:294) defines normalisation as 'a method of constructing files that eliminates the problems of repeating data elements and the insertion, deletion and update anomalies. The principle behind normalization is that file design should be based upon the relationships between the data elements rather than upon the application that the files serve'. He explains further that normalising a file involves defining the relationships between the data elements and 'decomposing' the data elements into a series of files based upon those relationships. There are at least three steps to this process, each of which results in a further refinement of what is called a normal form (Caswell, 1984:295).

McMurdo (1982:9), on the other hand, defines normalisation as an information science related activity, "a technique for determining the optimal logical design of an integrated database, based on the observation that certain groupings of entities exhibit better properties in storage operations than do other groupings containing the same data". He explains further that normalisation
techniques relates only to the logical or abstract
description of data rather than to its physical
representation that is concerned with various bits of
pointers, arrays, lists etc., which may be used to
represent information. By definition, normalisation is
therefore, a conceptual approach to the design of records
within files. Techniques and methods to represent data
into well structured records are employed to avoid
anomalies caused through the storage operations of
addition, update and deletion. McFadden & Hoffer
(1991:246) identifies the objectives of database design
as being structural validity, non-redundancy, simplicity,
shareability, extensibility and integrity. Normalisation
facilitates these objectives.

3.4.1.2 Steps in Normalisation
Normalisation theory provides a formula for a good
design. Relations or files are progressively upgraded
into higher normal forms (McMurdo, 1982:10) by removing
aspects of anomalies that may occur in the relations with
each successive step. Various authors have identified
specific steps or levels in the process of normalisation.
McMurdo (1982:10-15) distinguishes 5 levels of
normalisation while McFadden & Hoffer (1991:223)
distinguish 6 levels. For the purposes of this study only
three levels are discussed. According to McMurdо (1982:11), the first normal form (1NF) is actually the only normalisation required for a relational database. The higher normal forms, which are very much concerned with dependencies, all represent improved qualities in record structures. These higher forms are not strictly required for the implementation of a relational database. McFadden & Hoffer (1991:117) concurs with this view stating that 1NF relations can still have some undesirable maintenance properties. Most authors agree, however, that the third normal form is adequate (Ullman, 1982:237; Date, 1986:452) to serve as a basis for a good design. Stated briefly, a record is in the first normal form when all repeating groups are removed; in the second normal form when partial dependencies are removed and in the third (3NF) normal form when transitive dependencies are removed.

**First Normal Form (1NF)**

The first step in normalisation is to identify any repeating groups and remove them. 'A relation R is in first normal form (1NF) if and only if all underlying domains contain atomic values only' (McMurdо, 1982:10). An unnormalized relation is transformed into the first
normal form by splitting the relation into two, one for the repeating groups and the other for the rest of the data (McMurdo, 1982:10).

In the model presented above, the course number and name occurs twice. Such repeating groups must be removed and placed in a separate file (Course file as defined below) With the new structure, a new record is created for each new set of course and instructor details. To add a new reservation record, the Acc_No, Crse_No and Semester are required to uniquely identify a particular record in the Reserve file. Both files are linked by the Crse_No, which is unique. The Crse_No in the course record is the primary key (denoted by a +) while in the Reserve record it is a foreign key (denoted by a +). When a relation has more than one candidate key, one is usually designated as the primary key. Any key that corresponds to a key of another relation, even though it may not be a key of its own relation, is called a foreign key (McMurdo, 1982:10). With all repeating groups removed, the relation is in 1NF.

RESERVE (* Acc_No, Author, Title, *Semester, *+Crse_No) 
COURSE (*Crse_No, Crse_Name, Instr_Name, Instr_Phone)
This structure is more hospitable as it accommodates as many courses as possible. The course and instructor details are independent of the bibliographic and reservation details. The relationship between the records are established only at the time the item is placed on reserve for a particular course.

Second Normal Form (2NF)

1NF relations may however exhibit undesirable qualities in the performance of the basic storage operations of insertion, deletion and update thereby creating anomalies (McMurdo, 1982:11) and hence the need to consider higher forms of normalisation. McMurdo (1982:11) defines the second normal form as follows:

'A relation R is in second normal form if it is in first normal form and every non-prime attribute of R is fully dependent on each candidate key of R'.

The second normal form as defined above, introduces the concept of functional dependency. What is dependency or functional dependency?

Functional Dependency

Data items form aggregate or groups, for example, a bibliographic record consists of, inter alia, authors, titles and publishers. which define the bibliographic record. In ascertaining the relationships between the
data items, the database designer must identify which attributes are dependent on which item or items. The term 'functionally dependent' is defined as follows:

'Attribute B of a relation R is functionally dependent on attribute A of R if, at every instant in time, each value in A has no more than one value in B associated with it in relation R. The statement that B is functionally dependent on A means that A identifies B, that if the value of A is known, then the value of B can be found. In effect, it means that A can act as a key for B' (McMurdo, 1982:10).

'Dependencies are actual assertions about the real world' (Ullman, 1982:215). Ascertaining functional dependencies cannot be discovered automatically but requires a knowledge of the real-world that is being represented by the database (Cardenas, 1985:512). For instance, a supplier's address can be determined, given the supplier name. If one supplier has one and only one address, the supplier name functionally determines the address (Ullman, 1982:213). This is even true when each address has more than one supplier under it. However, when a supplier has two addresses then the name cannot functionally determine the address. This situation while not common, reflects the real world and has to be catered for in any design. With respect to the second normal form, attributes or data elements within a relation are required to be fully dependent on the key.
The reservations example cited above is not in 2NF unless all data elements are fully dependent on the key. In order to make all data elements fully dependent on the key in the Reserve file, the title and author, which are dependent on the Acc_No but not on the Semester or Crse_No which also constitute the key, is moved to a separate Item file. Adding a foreign key, Acc-No, in the Reserve file yields three files in the second normal form as follows:

RESERVE (*+Acc_No, *Semester, *+Crse_No)
COURSE (*Crse_No, Crse_Name, Instr_Name, Instr_Phone)

Now the bibliographic and course details are recorded quite independent of any reservation processes.

**Third Normal Form**

'A relation R is in third normal form (3NF) if it is in 2NF and every non-key attribute is nontransitively dependent on the primary key' (McMurdo, 1982:12).

The third step in normalisation specifies that all non-key elements in a file be dependent only upon the key and not upon some intermediate element. In the course record, the phone number is dependent on the instructor's name which in turn is dependent upon the course. This kind of transitive dependency must be removed for the relation to be in the 3rd normal form. The anomaly is

113
corrected by introducing a separate Instructor file consisting of a name and telephone number. The course is linked to the Instructor file by a common instructor name. The new files are as follows:

RESERVE (++Acc_No, *Semester, ++Crse_No)
COURSE (*Crse_No, Crse_Name, +Instr_Name)
ITEM (Acc_No, Author, Title)
INSTRUCTOR (*Instr_Name, Instr_Phone)

This three step process completes file normalisation. With the above design, each of the entities, that is, Course, Item, Instructor as well as the reservation process (Reserve entity) are defined separately, allowing changes to these records to be made independently. For example, the instructor’s name appears only once in the instructor file, and the telephone number needs be changed once rather than for different instances as in the original Reserve file. Separate files means that the respective details form a permanent part of the database, thereby avoiding any deletion anomaly. For example, reserved items may be deleted without affecting the status of the item, instructor or course entities. The insertion of a new course for instance does not depend on at least one bibliographic item being placed on reserve (insertion anomaly). The files are well designed and are linked logically. From the lowest level,
the instructor records are linked to the course file. The course and item records are linked to the Reserve records by their unique keys to provide a unique reservation record for a given semester.

3.4.1.3 Overnormalisation and Denormalisation

Through the process of normalisation, large files are reduced to smaller manageable ones; the principle of normalisation when applied to computerised files yield beneficial results by preventing substantial re-design, allowing for increased sharing of files on which a variety of applications may run (McMurdo, 1982:16). According to McMurdo (1982:9), the advantages accruing from fully normalised data structures include ease of use, flexibility and ease of implementation. A database that is normalised can be restructured in the face of new additions and changes implemented without the need to rewrite applications (McMurdo, 1982:16). There is however a danger of overnormalisation and the need to denormalise the records within the files.

According to Date (1986:461), the objective of normalisation is to reduce redundancy. Once dependencies and repeating groups are removed, normalisation should stop. It is possible to reduce the table, EMP(*Emp,
Ename, Dept, Salary) into the tables as shown below. The tables show dependencies between the employee and each employee, his department and his salary.

EN(*Emp, Ename)
ED(*Emp, Dept)
ES(*Emp, Salary)

There is little to be gained from such further reduction. The following is a trivial example showing dependencies. In the Issues system, three records are as defined in chapter 2 of this thesis. The activity (namely issuing a book), is represented in the E-R model by a diamond (see Fig 3.4). If the relationship is translated strictly in terms of the E-R model into a relational model, the resulting relations would be as illustrated below. A new Issues record is introduced to record the activity of issuing.

BIB (*Brn, Title, Author, Class, Note)
ISSUE (**Borwr_no, *+Acc_no, Ddate, Od)
ITEM (**Acc_no, Price, +Brn)
BORROWER (**Borwr_no, Name, Tel, Add, Status)

There is little point in such normalisation as the activity of issuing is invariably linked with the item itself. Denormalising the issues and item relations therefore leads to combining the relations into one relation as reflected by the following relation:

ITEM (**Acc_no, +Brn, Price, +Borwr_no, Ddate, Od)

116
According to Date (1986:462-3), a fully normalised database may be denormalised for the purposes of retrieval and update. He asserts that if two items of information are very frequently retrieved together and are only very infrequently updated, as a design guideline, denormalisation is acceptable if there is a clear advantage for doing so.

3.4.2 The Entity-Relationship Model (E-R Model)
Various data modelling strategies including, binary relationship models, conceptual graphs, deep structure sentence models, entity-relationship models, object-role models, semantic nets, set theory models and the standard relational, hierarchical and network models are used to represent data (Pangalos, 1989:25). The E-R model is the most commonly used model, based on the concepts of entities and relationships among entities. There is no standard E-R notation (McFadden & Hoffer, 1991:106) and this characteristic makes the model ideal for use in this study as it is amenable to additions and changes.

Entity modelling begins with the identification of entities and their relationships, which are the basic elements in the model. Entity modelling is a technique which enables the classification and arrangement of data
and their relationships into useful categories. It involves identifying entities, their relationship with other entities and mapping them appropriately.

The following definition of an E-R model is given by Daniels & Yeates (1984b:107): 'Entity models illustrate real-world relationships between different data items inside the system'. They define the entity model as a logical data model which has three basic components, namely entities, attributes and relationships (Daniels & Yeates, 1984b:119). Agosti et al. (1989:312), on the other hand, regard only the entities and relationships as basic to the E-R model. Atre (1980:4) defines a conceptual model (the E-R model is a conceptual model) as 'An inherent model of the entities with data elements representing them, together with the relationships interconnecting the entities'. From the name of the model, the entities and their relationships are the two main constructs of the model.

3.4.2.1 Entities
According to McFadden & Hoffer (1991:18), 'An entity is a person, place, object, event, or concept about which the organization wishes to record data’. From this definition 'An entity is a thing that exists and is distinguishable'
(Ullman, 1982:12). For instance each ant can be an entity if features that distinguish one ant from another can be ascertained (Ullman, 1982:13). Examples of entities are cars, buses, books and authors, each having their own properties and attributes. While some entities are concrete, for example, employees, computers, buildings, motor vehicles, others are abstract, for example, bank accounts, airline flights and sale of goods (Mcfadden & Hoffer, 1991:80-81).

The key to defining an entity is its distinguishing and different properties or attributes (Ullman, 1982:13). For example, for the two entities, authors and publishers, the first may consist of a surname, initials and dates of birth and death, while publishers may consist of a place, publisher and date of publication. An entity must have at least one attribute that distinguishes it from other entities in that class. All entities of a particular type can be grouped together into one class category or set, for example, all cars, all persons, all books etc. McFadden & Hoffer (1991:81) defines an entity class as ‘a collection of entities that have similar characteristics’, for example, customers, students and patients.
The unique property of an entity is called an identifier or key for that entity, which may consist of a combination of one or more attributes (Ullman, 1982:13). The identifier functionally determines other attributes in the entity set, for example, in the Bib record of the issues system, the Brn identifies the record uniquely. Given the Brn, the title, author and classification number can be determined unambiguously.

Entities themselves may form part of a larger entity, for example, entities like cars, buses, trucks can be grouped as transportation vehicles and are themselves classified and distinguished by different make of cars, for example, Opel, Ford or BMW. The identification of both entities and attributes must be relevant in context. For example, a person’s weight and height is relevant in a health club but not necessarily in the workplace.

3.4.2.2 Relationships
Entities do not exist in isolation but are associated with one another to form a network of entities. The associations are termed relationships. Atre (1980:85) defines a relationship as 'a mapping or linkage between two sets of data'. It can be one-to-one, one-to-many or many-to-many.
One-to-One Relationships: 'A given value of A has one and only one value of B associated with it: That is, if we know the value of A, then the value of B is implicitly known' (McFadden & Hoffer, 1991:86). In a one-to-one relationship, for each entity in either set there is at most only one associated member in the other set (Ullman, 1982: 15). Examples include a one to one relationship between a husband and a wife or a departmental head and a department. These relationships are mapped as follows:

```
DEPT-HEAD  DEPARTMENT
HUSBAND    WIFE
```

One-to-Many Relationships: 'Each value of data item A is associated with zero (if so specified), one, or many values of data item B. However each value of B is associated with exactly one value of A. The mapping from B to A is said to be many-to-one, since there may be many values of B associated with one value of A' (McFadden & Hoffer, 1991:88). Examples: one borrower borrows many books, a Chief librarian manages many Library Staff. These relationships are mapped as follows:

```
BORROWER  BOOKS
CHIEF-LIBRARIAN  DEPARTMENT
```
Many-to-Many Relationships: 'Each value of data item A is associated with zero (if so specified), one, or many values of data item B. Also, each value of B is associated with zero, one, or many values of A' (McFadden & Hoffer, 1991:88). Examples: authors and their works, departments and books purchased. These are mapped as follows:

AUTHORS ←→ WORKS

DEPARTMENTS ←→ BOOKS PURCHASED

Using the above mapping techniques, questions like who wrote what books are answered by following the relationships or links defined between the entities in the set.

Entity Supertypes and Subtypes

Sometimes it becomes necessary to divide an entity into subclasses, for example, to distinguish an employee who is a salesman receiving a commission from others who are not. The property (commission) is not applicable to non-salesmen. This situation gives rise to two entities, an Employee entity with general characteristics (for example, employee number, age, sex, salary) and a Salesman entity that has unique employee properties, for example, members receiving a commission. In such cases,
the specific properties are aggregated into a separate entity, so that properties not applicable to salesmen are eliminated from the record (Date, 1986:457). Employees, with an employee number, job, salary and commission are represented by the entities and attributes as shown in Figs 3.2a and 3.2b.

<table>
<thead>
<tr>
<th>Emp_No</th>
<th>Job</th>
<th>Salary</th>
<th>Emp_No</th>
<th>Commission</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Salesman</td>
<td>80K</td>
<td>E1</td>
<td>25K</td>
</tr>
<tr>
<td>E2</td>
<td>Accountant</td>
<td>100K</td>
<td>E4</td>
<td>40K</td>
</tr>
<tr>
<td>E3</td>
<td>Programmer</td>
<td>25K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E4</td>
<td>Salesman</td>
<td>85K</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig 3.2a Employee Entity
(Date, 1986:457)

Fig 3.2b Salesman Entity

The Salesman characteristics are extracted from both the Employee and Salesman entities using the common attribute Emp_No in both entities to find all the attributes pertaining to a particular salesman, for example, for salesman E4, the salary is 85K and the commission 40K. Every characteristic of Employee is automatically a characteristic of the Salesman entity. Salesman is a subtype of the Employee entity, and equivalently Employee is the supertype entity of the Salesman entity type.

'Entity type B is a subtype of entity type A (and entity type A is a supertype of entity type B) if and only if every instance of B is necessarily an instance of A.'
(Date, 1986:458). The subtype-supertype relationships can be represented by means of a type hierarchy as shown in Fig 3.3a.

![Diagram of the type hierarchy]

**Fig 3.3a**  
**Hierarchy Representation (ISA)**  
(Date, 1986:458)

Type hierarchies are known by various names, for example, generalised hierarchies (equivalent of supertype), specialised hierarchies (equivalent to subtype) and ISA hierarchies (expressed in terms of the relationship between the supertype and subtype, that is, 'every Salesman 'is a' Employee'. An ISA relationship (read A 'is a' B) is a relationship that points to itself, for example, mothers being part of the class persons, but is simultaneously a special type of person. The relationship is depicted by an arrow pointing to itself (see Fig 3.3b above).
Specific rules and conventions are employed to reflect the semantics of the E-R model. In modelling entities and relationships, a rectangle is used to identify an entity, a diamond, the relationships and an ellipse, an attribute. 'Many' associations between entities are represented by the 'crow's foot' and a 'one' association by a line with 'no feet'. Associations do not represent associations between individual data items but represent the lines connecting the whole boxes (entities), not just the data items near where the lines meet the boxes. The example on issues cited in chapter 2 of this thesis, is represented by the E-R model as shown in Fig 3.4. For each Bib entity there are many Items but for each Item there is only one Bib entity. Similarly, for each Borrower many Items are borrowed but for each Item there is at most one Borrower. The relationship 'Borrows' has a property of due date (Ddate) and overdue status (Od), each item borrowed is linked to one borrower.

![Diagram](image)

**Fig 3.4 E-R Model For Issues**
3.4.2.3 Value and Limitations: E-R Model

The E-R model serves as a tool between various users to organise, visualise, plan and communicate ideas. According to Ullman (1982:12) 'the entity-relationship model does an adequate, albeit imperfect job of modelling real-world situations... however if we view the structures defined in the entity-relationship model as conceptual schemes we shall not be grossly deceived'.

Entities, being conceptual schemes or logical views of data, are abstract and hence hardware independent and therefore have a more stable existence than records and fields which are implementation dependent. The conceptual model changes when the 'view' of the data changes while in the implementation model, with each new implementation of a DBMS. However, the use of a two level modelling process, one for abstraction and another for implementation is a drawback in that two separate techniques are employed. Pangalos (1989:26) predicts that in the future DBMS will include a component for handling a common database view that allows conversion of the conceptual model into convenient data structures and constructs.
The E-R model is only designed to map the real world while the particular DBMS models, that is, hierarchical, network or relational model are 'more closely tuned to the needs of the database system designers to deal with efficiency issues' (Ullman, 1982:1). It is normally the hierarchical, network or relational model rather than the E-R model that is used in database systems even though the E-R model is generally applicable to all three data models. The next step in the design process is implementation.

3.5 Implementation Design
Implementation design is concerned with mapping the conceptual data model into an appropriate database model, namely, the network, hierarchical or relational data models (McFadden & Hoffer, 1991:303) so that the data can be processed by a particular DBMS. Specific data definition languages for the DBMS are used to define the schema (McFadden & Hoffer, 1991:170). The schema consists of all the file, item, record and view definitions while its subschema, corresponds to the various views or subset of data for a specific application. The relational model, with specific reference to Dbase III+, is the implementation model under consideration.
The major inputs to the implementation design are the conceptual data model, the DBMS characteristics, and user processing requirements, for example, usage patterns, response times, backup and security parameters (McFadden & Hoffer, 1991:304). The Dbase DBMS characteristics have already been discussed in the previous chapter of this thesis. Hence two important aspects remain to be discussed namely, translating the E-R model into a corresponding Relational model and translating the user processing requirements in terms of the rules of the DBMS itself. A discussion of usage patterns, response times, backup and security while important cannot be accommodated in this study.

3.5.1 Steps in Implementing the Relational Model
A simple translation of the entities into a corresponding relational model requires that an entity set be represented by a relation, each tuple representing one entity in the entity set. The relationship itself, is translated into a relation (Ullman, 1982:21-22) thereby achieving a uniform method of representing entities and relationships. The final result is an integrated set of relations, provided that the relations have been normalised and primary and foreign keys defined, accordingly. Each attribute of the entity occupies a
column within each tuple. The identifier of the entity becomes the primary key of the corresponding relation which is a unique key for each instance of the entity in a table. A relationship key (or foreign key) on the other hand, has the property of linking one or more tables.

In transforming one-many relationships from the E-R model to the corresponding relations, a foreign key is placed in the relation that represents the entity on the many-side of the relationship, for example, one Borrower borrows many Items is reflected by the foreign key Borwr_no in the Item table. To transform many-to-many relationships, a new relation is created, consisting of the composite key of the two relations participating in the relationship (McFadden & Hoffer, 1991:247). For the set of keys T1, T2, T3 for titles (File T) and A1, A2, A3, A4 for authors (File A)\(^{10}\), the many-to-many relationship is represented below (File TA). The example illustrates for instance, T1 has authors, A4 and A2 while author A4 has written books with titles T1 and T3, which is in effect a many-to-many relationship between the titles and authors.

129
Fig 3.5 Files Showing Many-to-Many Relationships

In transforming ISA relationships into relations, the following strategy is recommended:

1. Create a separate relation for the class and for each of the subclasses.
2. The table (relation) for the class consists only of the columns or attributes that are common to all subclasses, plus a subtype identification column.
3. The table for each subclass contains only its primary key and the columns unique to that subclass (McFadden & Hoffer, 1991:215).

The example on Salesmen (Fig 3.2) follows the above rules. The complete salesman record is effected by a join of the subclass with the class relation.

Repeating groups are not allowed in relational systems. Only one value per column is allowed. To overcome repeating groups, a new relation for each repeating element is created. A link to the 'owning' relation is
established using a common key. This has been illustrated in the Reservations example, where the course details were isolated after applying 1NF and placed in a separate Course relation and a link established to the Reserve relation by a common course number, Crse_No.

3.5.2 Specifying Requirements and Database Integrity

During requirements definition it is important to identify constraints that would maintain database integrity. Three types of integrity constraints are identified: domains, referential integrity constraints and other business rules. Domain constrains are rules that define allowable data types, lengths, formats, and values for each attribute. Referential integrity constraints are rules that ensure the validity of references for an occurrence of one entity into another entity. In considering the rows of one table and the rows in another table, the attributes used as common attributes in a join, for instance, must have the same type and length. Business constraints ensure the integrity of a data item value in one table, given one or more data item values in the same or related tables (McFadden & Hoffer, 1991:202). Given a current date (Cdate), the status of Od is consistent with the due date when the over program mentioned in Chapter 2 is run.
With reference to the issues system, domain constraints used are primarily characters, the Brn, Accession numbers, Borrower numbers are positive numbers. The status has values 1, 2 or 3. Telephone numbers are limited to 7 characters, a set of integers. Borrower names are extracted from the Durban telephone directory, which serves as the domain for the name, address and telephone numbers.

For the Item record, the Price ranges from zeros cents to R999.99 while the Ddate is a valid date in 1993 or 1994. The system is hypothetically assumed to have begun in 1993. The classification number is, for the sake of convenience, reduced to a 3 character Dewey number, The Dewey Decimal System, hence forming the domain for the classification numbers.

With respect to the integrity rules, the 2 general rules outlined by Date in the previous chapter (section 2.4.3) apply. In applying the two rules, the first pertaining to the primary key and the second to referential integrity, the following conditions hold for the issues system.

1. The Brn, Acc_no and Borwr_no are unique for the
Bib, Item and Borrower files respectively.
2. The foreign keys +Brn and +Borwr_no in the Item table must have an existing Bib and Borrower tuple in the respective records with the foreign key matching the primary keys in length, type and values extracted from the same domain. However, when no foreign keys exist, the values are set to zero. In a fully relational system, such a field should not exist at all. In Dbase, 'undefined' or null values are represented by zeros if numeric and blanks, if characters.

3.6 Physical Design
Physical design is the last stage of the design process and follows the implementation design phase. Physical design goes beyond a simple translation of the entities and relationships into relations (logical design). During this phase, the logical database structures of the implementation phase are mapped to physical storage structures such as files and tables. Indexes are specified, together with considerations of access methods (for example, sequential, random, hashed, relative), record blocking, partitioning of files and other physical factors. Important issues relate to data volume and usage; cost of storage; choice of file organisation (for
example, sequential, indexed sequential, direct access files); types of indexes to be used and the overhead costs thereof. Equally important are the ways the logical record is to be implemented, for example, fixed length or variable length records, splitting of relations into one or more relations, use of data compression techniques etc.

The main objective of physical design is to achieve acceptable levels of performance, security and integrity and ensure efficient use of storage space (McFadden & Hoffer, 1991:303-304). A discussion of all the aspects mentioned above is beyond the scope of this study. However, two key aspects, namely, record structures and indexing are discussed, while the different file types are given some consideration.

3.6.1 File Organisation : Access and Processing
Data in files are comprised of characters which in turn are grouped together to form a field, an aggregate of which in turn forms a record. 'The basic component of many computer files is the record...each record comprises a number of data items which are organised into locations known as fields' (Rowley, 1980:42). One or more records, comprise a file. A file is a collection of related and
usually similarly constructed records treated as a unit, for example, a list of borrowers (Hunter, 1985:xv). Records are stored in one order only, but may be accessed in more than one order using different access methods thereby imposing a new or different order from the physical order. Different types of files and file accessing techniques are therefore identified.

Data in files are organised physically in two basic ways, sequentially and randomly. In the first instance, the records are stored in physical sequence, next to each other, while in the latter case, the records are not in a particular order (O’Brien, 1985:385). Random file organisation is usually termed direct or nonsequential method of file organisation.

However, various methods have been developed to access records stored in files. These file access methods include a sequential access method and an indexed sequential access method for files organised sequentially. A key transformation method, the linked list method and various other indexing methods are used for files using a random method of organisation. In practice, however, file access methods are frequently called file organisation methods, for example, an indexed
sequential file access method (ISAM) which accesses a sequential file using an index to provide random access is regarded as having an index sequential file organisation (O'Brien, 1985:386).

**Sequential File Organisation and Processing**

In a sequential file organisation, the physical order of records in a file are stored in a predetermined sequence. Records are arranged in a specific order according to record key. Sequential organisation requires that all records are sorted into a proper sequence before processing (O'Brien, 1985:387-388). Records cannot be accessed randomly by key. To locate a record, a file is searched from the beginning. The method is too slow to handle large files and applications requiring immediate update and response. Random access provides the solution to get to a desired record quickly.

**Random File Organisation and Processing**

In a random file organisation, records are stored in a random or nonsequential manner, that is, in no particular sequence and keys, pointers or indexes are used to locate data. In such an organisation, updates and processing provide immediate responses, unlike sequential files,
that require the data to be sorted first and then stored in sorted order. There are a number of ways to organise random access files.

**Key Transformation Access Method**

In a key transformation method, a hashing or transformation algorithm is used to compute the address of a record. The algorithm, is applied to the key, which computes to an address for that record. In this way, given the key, a record can be randomly stored and accessed, thereby providing speedy access. However, the technique has a drawback in that, one or more keys may compute to the same address, called collision (O’Brien, 1985:389-390).

**Indexed Access Method**

An indexed sequential file organisation is similar to sequential file organisation, except that an index is imposed on the natural, physical order of the records. ‘An index ... is a file in which each entry (record) consists of a data value together with one or more pointers. The data value is a value for some field of the indexed file (the indexed field)’ (Date, 1981:38). The
pointers identify records in the indexed file that has data values which match the data values in the index entries.

Indexes permit records to be accessed sequentially as in sequential files but in addition permit random access. Indexing files is a faster method than sorting and require less space that several sorted copies of a file. However, they are slower than direct files. This is due to the need to access an extra index file which is stored in secondary storage devices (for example discs) (O'Brien, 1985:390-391). While an index speeds up retrieval, it slows down update (Date, 1981:38). Index files, have additional overheads, for example, cost of creating, storing, and maintaining the indexes. In addition, large amount of disk space is required on secondary storage devices to store the index.

<table>
<thead>
<tr>
<th>Record Key (employee number)</th>
<th>Record Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>28541</td>
<td>101</td>
</tr>
<tr>
<td>35879</td>
<td>102</td>
</tr>
<tr>
<td>47853</td>
<td>103</td>
</tr>
<tr>
<td>50917</td>
<td>104</td>
</tr>
</tbody>
</table>

**Fig 3.6 Example of index**

(O'Brien, 1985:390)
Inverted File Access Method

This form of file organisation uses an index or list of addresses or keys for all records having the same property, arranged in key sequence. An inverted index may be tailored to suit specific requirements, for example, grouping of employees between the ages of 18 to 25, 26 to 30 and 31 to 35 (see Fig 3.7). A list of record addresses is available for each of the age groups. Keeping a list of keys related by some common criteria allows efficient processing, but this is done at the cost of setting up and maintaining the inverted indexes, which can be very large (O'Brien, 1985:392-393). With respect to bibliographic records, a DBMS should allow a single index to be created for individual elements or subfields (terms, keywords or words) of a field (Armstrong, 1984:198). DBMS do not as a rule, have this capability. Indexes may be declared on one or more fields or parts of it, but indexing on keywords or terms is not a standard feature of DBMS.

<table>
<thead>
<tr>
<th>Record Address</th>
<th>Employee Number</th>
<th>Age</th>
<th>Age</th>
<th>Record Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>28541</td>
<td>43</td>
<td>18-25</td>
<td>104, ...</td>
</tr>
<tr>
<td>102</td>
<td>35879</td>
<td>27</td>
<td>26-35</td>
<td>102, 103,...</td>
</tr>
<tr>
<td>103</td>
<td>47853</td>
<td>32</td>
<td>36-45</td>
<td>101, ...</td>
</tr>
<tr>
<td>104</td>
<td>50917</td>
<td>24</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig 3.7 Inverted File Organisation
(O'Brien, 1985:393)
When compared to the above categorisation of files and indexing techniques, Dbase has limited options to organise files and indexes. Dbase files are essentially sequential files with the option of declaring indexes making them indexed sequential files. When the record number is used to locate a record, the files are used as direct access files. However, use of relative addressing is hazardous, considering the primary key changes when the files are packed and records are deleted. Packing, deletes all unwanted records and reassigns record numbers in increasing sequence. No gaps are tolerated in the record sequence.

3.6.2 Records
At the physical level, a logical record may be split into one or more records or one or more records may be combined to form one physical record, that is, in the first case an entity is split while in the second case one or more entities are combined. The relationships or links are translated into pointers to records or blocks, the pointer to a block can be its absolute or relative address. Within a block, the record is located using the header which points to where the record is located in the block (Ullman, 1982:37). The resulting stored record
differs from its logical counterpart in that the stored records include, in addition, pointers to other records and other overhead data such as record length indicators, record identification numbers and optionally, field identification numbers.

The format and content of each record, at the physical level, is defined using a data definition language, and this definition is available to the DBMS. The DBMS takes care of how the files are organised and what the record content should be. From the above it is observed that the files and indexes undergo a transformation totally 'independent' of the conceptual model (E-R)\textsuperscript{12}. Once the entities and relationships are translated into the corresponding record types and linkages or record types expressing the relationship, the file manager handles the insertion, update and deletion of records as specified by the host language or Data Manipulation languages that updates the database.

3.6.2.1 Record Structures
Irrespective of the type of file used (hashed, sequential, sequential indexed etc.), files are generally organised into well defined record structures. It is possible to regard a file as a stream of characters with
no specific format or structure. The word format conveys the notion of a formalised framework or structure to hold data of varying content. Also implicit in the term format, is the idea of a set of rules or conventions governing the representation of data within the format (Gredley & Hopkinson, 1990:44). The terms record format, record structure and readable format are used interchangeably.

Basically two types of record formats are prevalent in DBMS, the fixed length format which has a pre-defined length, and the variable length record format where each record is of variable length. When considering record formats three elements are considered, the basic structure or framework of the record; the field names or attributes and the record content or data (Hunter, 1985:30).

3.6.2.1.1 Fixed Length Record Format
When each record in a file has a given length, it is called a fixed-length record. Every fixed-length record has the exact same format, the same fields occupying the same relative byte positions within the record (Fosdick, 1981:32). Dbase dbf files are essentially sequential or indexed sequential files of fixed field and fixed record
lengths. A pointer of 10 characters is used to represent a memo field, the memo field is used to accommodate variable length data stored in an associated dbt file. In the Borrower file each of Borwr_no, Name, Tel, Add and Status, are of fixed length and their positions are well defined within the record structure. Given such a layout and structure, the computer program is able to read and manipulate each record. In a DBMS like Dbase, the position is known to the DBMS and the user need only specify what fields require listing, manipulation or retrieval and the DBMS locates the field, irrespective of the position of the field in the record.

The fixed length format is suited to many applications where the fields are of fixed length and the lengths of variable data are within reasonable limits. Bibliographic data are not readily adaptable to fixed length formats because they have unpredictable lengths. Library catalogue records are invariably variable length in nature. The details such as authors, titles and publisher have varying length and the use of fixed length records is not economical for large databases. Further, the fixed length format is not suited to applications involving optional elements in the record structure, for example, notes in a bibliographic description or where
the structure requires greater flexibility in terms of number of repeatable elements, for example, a title, variant titles, cover title etc. Hence specific structures or formats to accommodate variable length records are required.

3.6.2.1.2 Variable Length Record Formats
Two types of variable length record format, the delimiter and directory formats are distinguished and identified. Combinations of both the delimiter and directory formats are, however, possible. Both the formats provide a structure and basis to identify each data element or field within the format.

Delimiter Format
The delimiter method for declaring variable-length records involves specifying one or more reserved characters to serve as delimiters. Delimiters may be chosen to mark the end of a field and the end of a record, each of which are different (Fosdick, 1981:39). The key to the format is that the delimiters chosen must never occur within the data fields. A delimiter must be a unique, reserved character, otherwise a field or record will be incorrectly perceived by the computer program as having occurred earlier than was intended. Needless to
say, this would cause serious program malfunction and inaccuracies in processing (Fosdick, 1981:40). Such confusion and ambiguity is resolved by defining the structure independent of its content.

In the following examples, variants of the delimiter formats are illustrated for a set of five fields, Record Number, Author, Title, Publisher, and Date, in that order. In example one, the content is implied by its order (the delimiter is '@'), content designators are absent. Using fixed positions implies that, an optional element must still have a place-holder for the field. In the example, the author is null and is enclosed by consecutive delimiters '@@'. The limitation of the format is that the order of the elements must always be the same and cannot be swapped.

1. @1761@@Oxford Dictionary@OUP@1989
2. @011761@@03Oxford Dictionary@04OUP@051989
3. @1761?Oxford Dictionary@OUP!1989

In example two, three characters are used to identify the content. The first three characters (consisting of one character for the delimiter '0' and 2 for the numeric characters that follow) form part of the structure, the two numeric characters defining the type of field, namely, 01 for the Record Number, 02 for the author, 03
for the title, 04 for the publisher and 05 for the date. A request to list the titles, means that the data following the ‘003’ up to the next ‘0’ is searched (Gredley & Hopkinson, 1990:52). Example three, on the other hand, uses unique delimiters to separate the fields. Where unique delimiters are used, the delimiter functions both as a delimiter and an element identifier. In the above example, the ‘?’ indicates the title and the ‘#’ the publisher. Repeatability is allowed in the second method, for example, two 03’s can be used to represent the title and alternate title. The structure of examples one and three do not appear to allow repeatability. This is accomplished by introducing a sub-level identifier within each field. Example 4 and 5 illustrates the use of sublevel delimiters (the character ‘^’ is used) for the titles.

4. @1761@@Oxford Dictionary^Concise Oxford Dictionary
   @OUP@1989
5. @1761?Oxford Dictionary^Concise Oxford Dictionary
   @OUP@1989
6. @1761@@TOxford Dictionary^Concise Oxford Dictionary@OUP@1989

In both the examples 4 and 5, no distinction is made between the title and title proper, for instance. Specificity is achieved, by using indicators to qualify the titles. An indicator of T for the title and A for
additional title is used (illustrated in example 6). No matter what format option is eventually chosen, the rules and the algorithms extracting the data must be defined explicitly and unambiguously. The position of each field and their names must be known to the DBMS.

Where the length of the records are not too large and the number of fields not too many, the delimiter method is ideal, despite the high processing speeds required by the format to extract each field. In the delimiter format, the field positions are defined relative to the other fields in the record. Each time a field is required, the entire record is processed and the field extracted. A more efficient processing technique would be to extract each field without processing the entire record. In such a format, an index or directory is used to point to where the data is located.

**Directory Format**

A directory is a guide or index to the contents of a record. For every entry in the directory, there is an associated data field, the index pointing to where the data is in the record. The starting position and length of the field is usually provided. Gredley & Hopkinson (1990:51) distinguishes between two types of directory
based formats, those using tags and those not using tags. A tag is a string of characters used to identify or name the content of an associated data field.

**Directory Format not using Content Designators**

00100000502202501761Oxford DictionaryOUP1990

In the above example, each of the 5 elements, are represented positionally by three characters, for example, record number starting at position 001, the next three characters for the author (000 implying the element is absent), the title starting at position 005, publisher at position 022, and date at position 025. The ‘0’ indicates the end of directory. The total length of the directory is 16 characters. This format is restrictive as no provision is made for the repeatability of the elements (Gredley & Hopkinson, 1990:51). This is due to the positions of the fields being fixed and there is, in addition, a lack of content designators. Where the number of elements in a record is large, the format leads to wastage of space in directory entries that are not used, that is, where the fields are blank and optional, as is the case with the author.
Directory Format with Content Designators

1001300540225025@1761Oxford DictionaryOUP1990.

The above example is the same as the one above, except that tags, indicating the content of the data are used, the tags forming part of the directory. When using the format, optional fields are omitted and repeatable elements can be included. Further the positions can be swapped as the tag defines the content. The tags defined are 1 for the record number, 2 for the author, 3 for the title, 4 for the publisher and 5 for the date. In the above example, there are no directory entries for the author as no author exists (Gredley & Hopkinson, 1990:51).

With the above format, the end positions have to be computed from the existing starting positions and field length, for example, to extract the publisher which starts at position 22, the next field position 25 is found by computing 25-22 (yielding a length of 3 characters). The directory entries are forced to match the field order sequence otherwise a more complex routine (program) is required to find the next highest field. The solution is to include the length of the field in a
directory. A simple Dbase routine like Substr (Field, Start-Position, Length) can then be used to extract the field. The MARC (Machine-Readable-Cataloguing) format makes provision for such a directory structure.

The MARC format contains three major segments, the leader, the directory and the data itself. In the SAMARC Format, the leader contains general information on the record. It includes the length of the total record, length of the indicator, record status (for example, new record), its type and class (for example, printed monograph). The leader is a fixed length field of 24 characters (SAMARC, 1980:1-7). The Directory consists of a three character tag, 4 characters for the length of the field and a starting position (5 characters, the displacement instead of the starting position is given which is one character less than the actual position). The total length of each directory entry is 12 characters. Entries are arranged in tag order. The data segment consists of bibliographic data such as authors, titles, series, notes and standard numbers as well as local holdings data and subject headings. Data is arranged into logical groupings or functional blocks, for example, 300 for notes, 400 for linking data and 200 for descriptive information.
not support fine-tuning of the physical database. The fixed length record structure of Dbase is a limitation. Technically, however, a physical mapping can be achieved by defining one or more physical records per logical record, but this cannot be defined in the Dbase dictionary. The variable length fields for notes, abstracts and similar variable length data are accommodated by memo fields. Each memo field is associated with a 10 byte pointer (that is, a piece of data to locate another piece of data) from the dbf to the Dbt file identifying one or more blocks in the dbt file. In this way variable length data are available via the associated dbt file, while still retaining the fixed nature of the dbf file. With each dbf file is associated one dbt file with different memo fields pointing to different blocks in the same dbt file.

Both implementation and physical design are restricted in terms of what the DBMS allows but not in terms of what the designer can implement, that is, ‘while the maximum size in characters for the physical record may be limited, the maximum size for logical records depends only on the ingenuity of the system designer and the needs of a specific application’ (Bordwell, 1984:160).
NOTES

1. A discussion of the E-R model is undertaken in section 3.4.2.

2. Rules preserve the integrity of a database.

3. Section 3.4.1.2 mentions some divergent opinions on the level of normalisation that is acceptable.

4. See discussion of functional dependency in section 3.4.1.2 which defines the key for a function.

5. The Call number is replaced by Acc_No in the original example cited by Caswell (1984:293-296).

6. See section 3.5.1 for the steps in translating the E-R model into an appropriate implementation model.

7. Note, this is only possible if the Item is at the one-end of the relationship.

8. Some entities do not participate in relationships with other entities and are stand-alone entities.

9. The technique used by Ullman (1982:19) is employed to show ISA relationships.

10. Other attributes are omitted and only keys are shown.


12. It is beyond the scope of the study to look at the
various fine-tuning techniques and options in physical design. The interested reader may consult Atre (1980:303-321).

13. In the former case customer numbers, International Standard Book Numbers, Telephone numbers apply. A person's address can be fixed to a suitable length, for example, 60 characters per line without much loss of data or truncation. But to use 255 characters for a title for instance, would lead to much wastage as titles vary considerably in length and often exceed 255 characters.

14. In small databases limited data loss due to truncation and limited space wastage is acceptable. In large databases this may not be acceptable.

15. An extensive discussion on the various MARC formats is provided in Gredley & Hopkinson (1990). The present study places emphasis on the various formats and structures, showing their limitations and advantages.
CHAPTER 4
MANUSCRIPT CATALOGUING

4.1 Introduction
Libraries acquire, organise and arrange documents for use by their clients. The different kinds of materials, namely, books, serials, maps, music sheets, audio-visual material and manuscripts are organised and arranged in a systematic manner so that they can be retrieved efficiently. A systematic record of the holdings of a collection is the catalogue. Its purpose is to guide the reader in the identification and retrieval of documents. A catalogue is 'a list of books, maps, recordings, coins, or any other medium that composes a collection. It is arranged by alphabet, by number or by subject' (Wynar, 1980:633). It is a record of the documents in an institution, compiled and arranged in such a way that it guides the reader in the identification and use of documents. It is therefore a retrieval device of sources of information in an institution. Each item reflected in a catalogue, must however, be given one or more entries to describe and identify the item. 'The process of describing an item in the collection and assigning a classification number' is termed cataloguing. Descriptive
cataloguing, on the other hand, is 'the phase of the cataloging process concerned with the identification and description of library material and the recording of this information in the form of a catalog entry' (Wynar, 1980:634).

4.2 Objectives of the Catalogue

Charles Cutter set the following two fold objectives for a catalogue:

1. To enable a person to find a book of which the author, the title or the subject is known;
2. To show what the library has by a given author, on a given subject or in a given kind of literature (Malinconico, 1979:47).

In terms of the above two objectives the catalogue must show whether the library has a particular item or publication, by a certain author, title or subject. It must also identify the work represented by the item or publication and relate the various works of an author, the various editions and translations of the work (Malinconico, 1979:48). An equally important function of a catalogue is to facilitate the physical location of a specific item.
The catalogue is a means of recording and controlling the content of an institution. 'Bibliographic control consists of those activities that are necessary to create and organize records that identify and describe library material' (McCallum, 1980:376). This involves the cataloguing and arranging of the items and the resultant records for retrieval. One of the instruments by which bibliographic control is maintained is a catalogue. This effectively means that the items in a library are more than just a finding list but reflect a bibliographically integrated picture of the resources of the library. By representing an helpful order, the catalogue can more readily serve the needs of the user. For instance, the acquisitions librarian needs to know which books are in the library so that more copies can be ordered or an order placed for a new book. Management requires statistics on the cost and number of titles purchased in the library. Patrons need to borrow items from the library. These are among the many uses of a catalogue and it demands efficiency and organisation.

4.3 Need for a Cataloguing Code
Communication cannot take place without a shared language, and when it is desired that communication should take place unambiguously, standards become

157
essential. Standards are instructions or techniques for doing something uniformly. Part of a standard is to embody the communication process in a set of rules or codes. Standards in the way catalogues are displayed, described and accessed are essential to avoid repetitive solutions to the same problems and to simplify catalogue construction. The rules are designed to create order and reduce inconsistencies in catalogues. Cataloguing codes set out rules to perform the function of cataloguing. The rules set out various conditions, for instance what elements are to be included or excluded and how a catalogue entry is to be formulated and constructed. The main objective of codification is to avoid the wastage of time and effort by the cataloguer. Some major matters calling for codification are:

- the selection of essential information when constructing a catalogue entry. Within the AACR2R, for instance, decisions concerning what elements are to be included and the extent of detail are governed by the preferred level of description;
- the order and layout of the different elements in a description (AACR2R, 1988:14-15). These include any punctuations in the layout;
- the choice of main entry heading from other possible
entries. The rules set levels of precedence in
deciding which 'author' has primary responsibility;
- the form of the heading, for instance in the case
of authors known by different names or variant forms of
a subject heading;
- the nature and composition of a catalogue entry.
Besides specifying how the catalogue is to be
constructed, principles of entry are included, for
example, in the AACR2R, the principles of main and
alternate entry are explained (AACR2R, 1988:2);
- Determining the nature of relationship complexity.
This includes rules that stipulate micro and macro
level descriptions and mechanisms to link them. In
the AACR2R the use of added entries, content notes,
analytics and multi-level description are techniques
described to achieve this (AACR2R, 1988:299-302).

4.4 The AACR2R
The 1988 revision of the Anglo-American Cataloguing Rules
(AACR2R), is the latest version of the rules. The
beginning of the 20th century marked the beginnings of
Anglo-American co-operation. In 1902, the Library of
Congress and the American Library Association (ALA) set
about creating a standard code. Britain joined the
American organisations in 1904, to develop a joint code.
The completed code appeared in 1908 in two editions, an American and British text (Malinconico, 1979:312) which were identical in wording except where there was a divergence of opinion (Brown, 1990:82). The next major phase of Anglo-American effort was the development of the 1967 Anglo-American Cataloguing Rules (AACR) comprising of two texts, an American and a British text. Unlike its predecessors, the 1967 rules gave a more systematic presentation of the rules and was based on specific principles. Both the 1908 and 1949 codes were enumerative, that is, they listed problems with solutions using rules (Brown, 1990:82). Subsequent revisions to the 1967 edition, namely, the 1978 (AACR2) and 1988 (AACR2R) editions saw significant changes, the most important being the incorporation of the ISBD as part of the rules; use of separate rules for different categories of materials and incorporation of the American and British texts into one single text.

4.4.1 Composition and Construction of a Catalogue Entry
A catalogue entry in AACR2R is divided into two parts. Part one consisting of a description and part two consisting of a heading (see Fig 4.1). The two parts of the entry correspond to the two parts of the code. The rules in part one is concerned with the physical medium
of the item being catalogued, to the level of detail
required for each element in the description and to the
analysis of an item containing separate parts (AACR2R,
1988:1-2). The description provides sufficient
information to distinguish one book from another.

Part two of the code deals with the construction of
headings. The heading consists of elements that
characterise a book, providing information on the author,
title and subject relationship to other books which are
useful but not essential for identification. The rules
deal with the organisational factors relevant to the
selection of access points and collocation of
bibliographic entries for all types of materials. The
chapters cover the choice of main and added entry
headings dealing with works of single authorship,
unknown, shared, changing, mixed and diffused authorship.
The second section treats matters pertaining to the form
of headings and uniform titles and includes rules for the
formulation of references (Byrum & Hinton, 1979:174-175).

The heading is either placed on separate lines above the
description or it is separated from the description by a
full stop and spaces. Where an entry begins with a title
proper either the title proper is repeated on a separate
line above the description or the description alone is
given (AACR2R, 1988:2).

Wynar, Bohdan S


Bibliography
Includes indexes
ISBN 0-87287-221-1

Fig 4.1 A Typical Catalogue Entry
(Wynar, 1980)

The cataloguing process begins with the determination of
the description of the physical object followed by the
selection and choice of main and added entries. The aim
of the two step process is to first establish a
self-sufficient set of descriptive data relating to the
physical object being catalogued (book, motion picture,
sound recordings etc.) and secondly, to add access points
such as the name, title, uniform titles and subjects to
retrieve the description (Gorman, 1978: 211-2). This kind
of distinction between an item that is object-based in
contrast to its relationship in a wider context to which
the object is related, goes beyond just retrieval, but
includes the collocation of individual items under one main entry, which is one of the many possible access points that may be constructed.

4.4.2 Cataloguing Principles: Item and Collection Levels
To catalogue an item means to describe it, to write out a bibliographic description. Before an item can be described, a decision must be made on what constitutes the item to be catalogued. An item according to the AACR2R Glossary is 'a document or set of documents in any physical form, published, issued, or treated as an entity, and as such forming the basis for a single bibliographic description' (AACR2R, 1988:619). The definition implies that while some entities are single documents, others comprise of two or more documents. If the item at hand is a single document, then there is no choice as to what constitute the item to be catalogued; the item at hand is the item to be catalogued. If, however, the item at hand constitute several physical objects, then the documents may be catalogued at two levels, and a choice has to be made on how the items are to be catalogued. Where an entire set has one overall title, and the individual documents have no title of their own, then the entire set constitute the item. This item is described under what may be called the set
description principle. However, when a multi-document set has a title for the set and distinct titles for each document, the entire set can be kept together under the set description principle, in which case a main entry for the entire set is constructed; otherwise each document is catalogued as a separate item, and a main entry constructed for each document. The former is termed the set description principle while the latter is called the document description principle. Commenting on the basic principles Hoffman (1983:395-396) states 'Only the two principles mentioned above, the set description principle and the document description principle, are possible, and that every publication cataloged, in all types of media, must be treated according to one or the other'.

4.4.3 Analytics

The set and item level principles sets out two broad levels for cataloguing manuscripts. Chapter 13 of the code outlines the possible options for the cataloguing of an item at one or more sub-levels. To achieve this an analytical entry is made for 'part of an item for which a comprehensive entry is also made' (AACR2R, 1988:615). This involves the process of analysis. Analysis is 'the process of preparing a bibliographic record that describes a part or parts of an item for which a
comprehensive entry has been made' (AACR2R, 1988:299). When a comprehensive entry for a larger work exists, the part is named either in the title and statement of responsibility area or in the note area or an added entry for that part is made. The added entry consists of the main entry or the title. The latter method is preferred when direct access to the part is wanted without creating an additional bibliographic record for the part (AACR2R, 1988:299-300). When the part is named in a note, the part is listed in a contents note consisting of the title or the name and title, for example 'Contents: Vol. 1. Plates - v.2. Text'(AACR2R, 1988:300).

If more bibliographic description is needed for the part than can be displayed in the note area, an 'in' analytic entry is made. The descriptive elements in the part consists of the title, statement of responsibility, edition, publication, distribution, extent of item and notes. This is followed by a short citation of the whole item in which the part occurs, the citation of the whole item beginning with an 'in' followed by the name and/or uniform title heading, title proper, statements of responsibility (when necessary for identification), an edition statement and publication details. The order of
the elements and rules applicable to analytics are also applicable to analytics within analytics, in which case the smaller item is mentioned first. Examples:

Miss Mapp / E.F. Benson. - 310p. ; 23cm.
In [Heading]. All about Lucia. - New York : Sun Dial Press, 1940.


Multi-level description is used when complete identification of the part and the whole item is needed in a single record representing a complete description. The description is divided into two or more levels, the description of the whole is given first, followed by a description of a group (of parts) and then the individual parts at the third level (AACR2R, 1988:302). Fig 4.2 illustrates a multi-level description.


166

**Fig 4.2 Multi-level Description (AACR2R, 1988:302)**

### 4.4.4 Alternate and Main Entry Approaches

The AACR2R caters for two approaches to cataloguing, the alternate and main entry systems. Lubetzky (1979a:18) clarifies the position regarding use of the words 'entry' and 'unit entry'. The word entry is used in two senses, entry as in main and added entry or entry meaning unit description. Title-unit-entry means 'an entry that begins with a title' (Lubetzky, 1979a:19). The title-unit-entry is not to be mistaken that the title is the main unit of entry, rather the unit begins with a title instead of a main heading. Whether an entry begins with the heading or title defines the type of entry. From this, it is inferred that in the main entry approach, the main heading commences the unit of entry, followed by the International Standard Bibliographic Description².

In the alternate entry system, the title is used as the unit of entry (called title-unit-entry) and consists of the ISBD only. The rationalisation for the
title-unit-entry is that generally speaking there is no common agreement on the choice of main entry and the form of a heading. The title on the other hand is certain. If the title is seen as the chief identifier then every other entry is added and this includes the main entry. 'The title-unit-entry concept is exactly the same, or will look exactly the same as the main entry. The only difference is the cataloger doesn't have to sit down and challenge himself, select one entry over the other, and say that this person is more responsible than another person for the work' (Lubetzky, 1979a:17). Cataloguing is reduced to a simple process of describing a book under its title, followed by a list of appropriate headings, from which the desired catalogue could then be automatically composed and reproduced by the computer (Lubetzky, 1979a:10).

Figure 4.3a shows a typical title entry without a heading using the alternate entry principle while Fig 4.3b shows a main entry with the title as an added entry. In both the approaches, headings may precede the basic unit of entry. In the alternate entry approach, even though the main entry heading is one of the access points, no attempt is made to distinguish it from other entries (compare Fig 4.3a and Fig 4.3c). In the main entry
approach, the heading is placed above the basic unit of entry giving the impression of a three level entry (Fig 4.3b) and when a title main entry is used, two levels (Fig 4.3c).


**Fig 4.3a Alternate Entry** (Wynar, 1980)

Introduction to cataloging and classification
Wynar, Bohdan S

**Fig 4.3b Added Entry using Main Entry Principle** (Wynar, 1980)

Introduction to cataloging and classification

**Fig 4.3c Added Entry for Alternate Entry** (Wynar, 1980).

Use of the Title-unit-entry approach has come under much debate. Clearly, the title-unit-entry represents a book under its own title, and without the main entry renders a catalogue as a list of books and not of works, as the related editions and translations are not brought under one main heading. The title-unit-entry utilises added entries to bring together items of a particular author,
but this means bringing together books, and not works of
the author. Using added entries only as in the alternate
entry approach obscures the main responsibility, for
example, when there are two or more authors. Use of added
entries does not mean that anyone of them is the main
entry. It has never been defined in the first place.

4.5 Main Entry and Main Entry Heading

The view that the catalogue is 'a simple list designed to
help one find a particular book in the library and
nothing else' was challenged by Sir Anthony Panizzi.
Panizzi's ideology was based on the following perceptions :
that the book sought by a person is really, most
frequently, not the object of his/her interest, but the
work contained in it is; that the work may be found also
in other editions, translations, or versions, published
under different names of the author and/or different
titles, some or all of which may be of equal or greater
interest to that person. For as Panizzi saw it 'A reader
may know the work he requires; he cannot be expected to
know all the peculiarities of different editions, and
this information he has a right to expect from the
catalogues' (Lubetzky, 1979a:7-8). This sharp distinction
between a book and the work paved the way for the concept
of the main entry and use of uniform titles.
A catalogue must in the end serve as an informative guide to the library's resources, for example, distinguish between, for instance, a text and a film version of Shakespeare's Hamlet. Hence, two levels of listing can be envisaged, the item and works level, the item level pointing to individual responsibility while the works to a higher level of responsibility to which the individual items relate. Lubetzky (1979a:17) remarks that 'Fixing the position of a work in the catalog makes it possible to have an integrated catalog, where under the entry of the work you have everything that relates to it'. For each item described only one main entry heading is supplied and this is supplemented by added entries. Where no main entry heading is distinguished, then each heading is of equal value (termed alternate entry headings) (AACR2R, 1988:2).

Ranganathan defines the main entry as 'Specific entry giving maximum information about a whole document. All other entries, specific or general, relating to the document in question, are derived from the main entry' (Carpenter, 1989:91). The AACR2R (1988:619) defines the main entry as 'The complete catalogue record of an item, presented in the form which the entity is to be uniformly
identified and cited. The main entry may include the tracing(s)' of all other headings under which the record is to be represented in the catalogue.

For the purposes of the study, the main entry consists of the ISBD and a main entry heading. Both components of the entry are sufficient to identify a record, the first component describes the item as completely as possible and the second component places the item in relationship with related records having similar characteristics. Other access points are excluded. Part of the reason is that added entries are totally absent in the sample data while main entry headings are not present in all examples in the sample. The added entry is not a necessary component in identifying an item. The main entry heading is optional and this is maintained throughout the study as many of the examples in the sample do not have headings.

Despite the present role of the main entry, the use of alternate entries, multiple permuted access and authority control mechanisms may weaken the role of the main entry in the future. To this effect Gorman (1978a:218) remarks

172
that 'It seems unlikely that the dominance of the main entry will survive into future machine-system-based codes of cataloguing rules'.

4.6 International Standard Bibliographic Description (ISBD)

The ISBD serves as a basis and standard for all AACR2R descriptions. It was not after the International Conference on Cataloging Principles (ICCP) meeting in Paris (1961) that descriptions was given attention. Before this meeting, code development concentrated primarily on the main entry and headings. Curwen identifies three developments that caused a re-look at descriptions after the ICCP meeting. Firstly, the Shared Cataloging Program launched in the United States in 1966 required that cataloguing take place in the countries that the publication occurred. The resultant shared resources (headings and description) for each item were made available to those participating in the program. Differences in cataloguing practice of the major bibliographic agencies of different countries were evident when entries were pooled together into one catalogue. Secondly, the use of computers in cataloguing for the purposes of exchange in machine-readable form meant that bibliographic data in machine-readable form
should be consistent and uniform and standards became critical. Thirdly, the ICCP generated a text of the principles agreed upon with full examples to illustrate them. The examples showed an embarrassing variety of descriptive practices, and International Federation of Library Associations (IFLA) felt that this was most undesirable (Curwen, 1991:75). Clearly, variations in standards make it difficult to distinguish the same item or the same edition of a work when there is a proliferation of editions and translations of the same work. Non-standard descriptions in one catalogue, even if under the same heading may appear as different items and editions.

The objectives of the ISBD were to aid three processes:
1. The interchange of bibliographic records from different sources,
2. The interpretation of records in different languages,

In terms of the first objective, the records from one source should in principle be usable in different systems and countries without changing the descriptions, even if
changes were to be made to the headings. Such standardisation is meant to facilitate exchange of information across national boundaries. Secondly, parts of a record should be recognisable and identifiable no matter unfamiliar the language of cataloguing. Thirdly, a standardised structure should enable machine-readability of the records (Curwen, 1991:76-77).

The above objectives directed efforts towards the development of a framework to accommodate all the descriptive data commonly required. The final version of the ‘International Draft Standard Bibliographic Description’ was presented, and formally published in 1971 and this was followed by a revision in 1978 (Milcetich, 1982:178-179). Further advancement was made when in 1976, the ISO’s Technical Committee 46 accepted the ISBD as an ISO standard (Milcetich, 1982:181).

4.6.1 Nature of the ISBD
Gorman (1978a, 213) states that the development of the ISBD(G) and the many individual chapters on specific types of materials, ensures that all library materials would receive equal treatment. The ISBD forms a standardised framework for such materials. The result of these efforts culminated in a consistent approach to
bibliographic description with regards to order and punctuation within the description regardless of the type of material being catalogued. The ISBD breaks away from the notion that data must be transcribed in title-page order, that is, it does not matter if the title proper follows or is preceded by the sub-title. The objective is consistency, not typographical design. Fig 4.4 illustrates an example of an entry in ISBD format.


Fig 4.4 Example in ISBD Format
(Curwen, 1991:77)

The ISBD is divided into eight areas divided as follows:
Title and statement of responsibility
Edition
Material (or type of publication) specific details
Publication, distribution etc.
Physical description
Series
Notes
Standard number and terms of availability.
Each of the eight areas is further subdivided into one or more elements, for example, the physical description is divided into the extent of item, dimension and other details. The ISBD is limited to describing the document and is shorn of all access points. It usually commences with the title and ends with the standard number area. Elements in the description are mandatory if applicable. Hence, the last element may be the physical description, series, note or standard number. The ISBD is meant to be independent and intelligible in the absence of any main entry heading. Pre-ISBD descriptions are not independent because they frequently lacked a statement of responsibility (Cathro, 1980:61).

The ISBD prescribes the use of punctuation as symbols to introduce or enclose the various areas and elements of the description (Curwen, 1991:77). Each area is separated by a ‘full stop space dash space’ or optionally each area is introduced on a new line when the ‘full stop space dash space’ is omitted from the area following the indentation. The punctuation differs from area to area but is consistent within each area. The esoteric punctuation marks are ‘a signal to the cataloguer as to what elements
they are examining' and that this aids the cataloguer in identifying the elements even if the language or the meaning of the words are not known (Lubetzky, 1979:16).

4.6.2 Levels of Description

Different kinds of libraries require different degrees of detail in their catalogue entries. In terms of Rule 1.0D, AACR2R recognises three levels of description. Cutter provided for short, medium and long descriptions in his cataloguing rules and the minimum, standard and full description in AACR2R correspond to Cutter's three levels, respectively. While the short entry gives the minimum information, the second level, gives all the data necessary for descriptions in medium to large libraries. The long entry, gives all the information that can be fitted into the standard framework and covers every descriptive element found in the rules.

The descriptive elements are to be considered as a minimum for the level. When appropriate, further information may be added to the description at any given level. Choosing a specific level as a point of departure additional elements are added for different types of materials in a consistent manner as determined by the policy set out by the institution. Hence the levels of
description allow institutions flexibility in their cataloguing policy (AACR2R, 1988:9). The members are not forced to adhere to impossibly high standards but are given a choice of different levels to choose from that is appropriate to their needs. Below is the general format for the first and second level ISBD formats. Each level specifies the order of the elements and symbols used. The third level includes the second level and elements such as the language and script of the description and items with several chief sources of information (AACR2R, 1988:15-17).

**Fig 4.5 First Level ISBD**
(AACR2R, 1988:15)

Title proper / first statement of responsibility\(^3\). - Edition statement. - Material (or type of publication) specific details. - First publisher, etc., date of publication, etc. - Extent of item. - Note(s). - Standard number.

**Fig 4.6 Second Level ISBD**
(AACR2R, 1988:15)
Today, the ISBD is incorporated into varying formats and cataloguing codes, for example the MARC and the AACR2R. The IFLA Working group resolved at a meeting in Brussels in Feb 1974, to use the ISBD as the foundation of the definition of functional areas for the MARC format (Avram & Rather, 1974:163). The ISBD as a standard has been adopted by and introduced into at least ten national and multi-national cataloguing codes among them being, the AACR2R, the German cataloguing rules, the Regln fur die Alphabetische Katalogisierung (RAK) and the Nippon Cataloguing Rules prepared by the Japan Library Association (Anderson, 1982:12).

4.7 Manuscript Cataloguing According to the AACR2R
Manuscripts and manuscript collections are found in many kinds of institutions; public libraries, historical societies, college and university libraries, research institutions and governmental libraries and archives. The AACR2R rules covers description and entry of all library materials (AACR2R, 1988:1). Part I, begins with a general chapter which contains general rules applicable to any form of material. Elements such as the place of publication, edition statements apply equally to all types of materials and are treated in detail in chapter 1 of the code. Elements requiring special treatment are
discussed under the relevant chapters for books, manuscripts, sound recordings, graphic materials, serials, maps, audio-visual material, computer files and music sheets. Separate rules for manuscripts exists within the broad framework of the rules in chapter 1.

Chapter 4 of the AACR2R, outlines the rules for manuscripts. These include materials of all kinds, including manuscript books, dissertations, letters, speeches, legal papers but excludes reproduction of manuscripts published in multiple copies. A manuscript or document is 'any text in handwriting or typescript (including printed forms completed by hand or typewriter) which may or may not be part of a collection of such texts' (Hensen, 1983:7). Examples of manuscripts and documents are letters, speeches, diaries, minutes, marked or corrected gallery and page proofs, manuscript books and legal papers (Hensen, 1983:7). A manuscript collection on the other hand, is 'a body of manuscripts formed by or around a person, family group, corporate body, or subject either from a common source as a natural byproduct of organic activity or gathered purposefully and artificially without regard to original provenance'. Besides the types of manuscripts listed above, collections contain correspondence, memoranda, business
records, photographs, maps, drawings, pamphlets, periodical tear-sheets and newspaper cuttings (Hensen, 1983:7).

'In the narrower sense of the word a manuscript is normally described as a 'hand-written or typed document'. When reference is made to a 'manuscript collection', however, a much broader meaning to the term is attached. In this context it can include items such as correspondence, diaries, financial documents, and even information carriers such as photographs, microfilms, films and tape-recordings. Books are excluded (NAREM, 1985:1).

A manuscript collection is not limited to personal papers only but can also include the archives of one or more organisations. For instance, a museum archivist may collect documents, photographs and furniture from the public while and also keep files, registers, and other items of the museum in an archive. While the former is collected at random, and sorted according to the taste and institutional requirements; the archive items are on the other hand, systematically accrued in the course of museum administration and are arranged according to the registry system of the office of origin (NAREM, 1985:1).
The AACR2R makes provision for the cataloguing of individual manuscripts and collections of manuscripts. Figures 4.7a and 4.7b illustrates examples of entries in ISBD format for an individual item and a collection. Again the rules and guidelines specified for all items apply to manuscripts. These include, the manner in which the entries are constructed and the use of the ISBD as outlined in the previous section. However, not all elements applicable to other material apply to manuscripts.

IN : Biographical collection of first Principal, 1930-1938
Part typescript, part manuscript

Fig 4.7a Individual Manuscript in ISBD Format
(Merrett, 1990:15)

Biographical collection of the first Principal, 1930-1938 / [John William Bews]
J.W. Bews was Professor of Botany from 1910-1938 and Principal of the University of Natal, 1930-1938
Analysed

Fig 4.7b Collections in ISBD Format
(Merrett, 1990:15)

The ISBD for manuscripts does not comprise of all eight areas mentioned in section 4.6.1 of this thesis. The Material or Type of publication details, series and standard number areas are excluded. The areas and elements that are relevant to manuscripts are shown in

183
Fig 4.8. The areas group elements into logical categories, relating like components, for example, title and its associated responsibility are kept together as distinct from the edition and its associated responsibility. Equally so, all details relating to the physical description, namely, dimension, extent of item and other details are kept together.

Title proper [general material designation] = parallel title : other title information / first statement of responsibility ; each subsequent statement of responsibility. . Edition statement / first statement of responsibility relating to the edition. - Date(s) of publication. - Extent of item : other physical details ; dimensions. - Note(s).

Fig 4.8 Second Level ISBD for Manuscripts
(AACR2R, 1988:15)

The study adopts the second level ISBD format and this forms the basis for the analysis of manuscript data. The elements within the areas are explored for analysis. However, first, an extended E-R model is developed to accommodate bibliographic data.

4.8 Extended Model for Bibliographic Elements

While the E-R model, which was defined and illustrated in the previous chapter of this thesis, provided the basic constructs for mapping entities and their relationships, these are not adequate. The model requires extensions
that reflect the complex nature of bibliographic records and the special nature of bibliographic elements. This aspect is explored.

The bibliographic entity can be equated to the main entry (defined under section 4.5 of this chapter) which includes a heading, descriptive elements and tracings. The tracings include access points for the statements of responsibility, series and subject headings. The ISBD is also an entity in its own right, having an independent existence and identifiable as an object. The elements in the ISBD, however, form a subset of elements in the bibliographic entity. Gredley & Hopkinson (1990:9) define a bibliographic record with reference to the ISBD as 'the sum of all the areas and elements... which may be used to describe, identify or retrieve any physical item (publication, document) of information content'. For the purposes of this study, the ISBD is an entity, defined as the sum of all the areas and elements used to describe and identify a bibliographic item. The definition incorporates elements identifying manuscript items and collections.
Date (1986:427-431) categorises entities into kernels, associations and characteristics types. Kernels have an independent existence 'it is what the database is all about', for example, the bibliographic entity. An associative entity or association is an entity whose function is to represent a many-to-many or (many-to-many-to-many, etc.) relationship among two or more entities, for example, a shipment is an association between a supplier and a part; each participant in the association can be a kernel, association or characteristic entity. The above two correspond to the entities and relationships in the E-R model.

The characteristic entity is an entity whose sole function is to qualify, describe or characterise some other entity (kernel, association or characteristic) but characteristic entities are existence-dependent on the entity they characterise. Entity B is said to be existence dependent on entity type A if it is impossible for an instance of B to exist while no other corresponding instance of A exists, for example, in the salesman example cited in section 3.4.2.2 of the thesis, cannot exist on its own independent of an employee entity as it shares characteristics of a larger (the employee) entity. A kernel, characteristic or association entity
can further designate some other entity, for example, an employee designates a corresponding department or a flight designates a corresponding airline (Date, 1986:427).

Bibliographic entities display characteristics described by Date. The ISBD is part of the bibliographic entity, the bibliographic entity forms the kernel entity and the ISBD is dependent on the kernel entity. However, the ISBD can also be considered independently with its own key, with numerous entities being dependent on it. The entity (ISBD) is characterised by titles, author statements, editions, etc. of which one or more descriptive elements are entities in their own right, for example, a publisher can be further subdivided into place, publisher and date(s) and can be perceived as an entity. The relational model does not allow complex elements and repeating groups (requirement of the first normal form) and the only way to by-pass these limitations is to split the complex entity into two simpler entities. Both these criteria, trigger numerous entities within the ISBD, for instance the edition statement consists of an edition and zero, one or more statements of responsibility. By normalising the elements all repeating groups are placed in a separate entity. While it is difficult to perceive
of the edition area as an entity on its own, yet for the purposes of analysis, the rules for normalisation shall be applied\textsuperscript{5}.

The entities within the ISBD can be said to characterise the ISBD as well as designate it. An interesting fact about designations is that while they map into separate entities, in a database they do not normally map into separate records (although there is nothing to prevent this); but forms a field (Date, 1986:438)\textsuperscript{6}. In the study, characteristic entities are modelled as in Fig 4.9. A weak entity is identified by strong vertical lines on either side of the rectangle. This is an adaptation of Chen's 'weak entity' (Chen, 1976:18-19). Chen, the originator of the Entity-Relationship model, distinguishes between three categories of entities calling them regular, weak and relationship entities. This corresponds to the categorisation identified by Date (1983a:230).

\textbf{Fig 4.9 Weak or Characteristic Entity}

188
In a catalogue entry some elements are mandatory, for example, title and physical details. Other elements like the GMD (General Material Designation) are optional. However, when a specific level is chosen, some elements are mandatory if applicable (MA), for example, a statement of responsibility for the edition is mandatory at level two, if it appears in the document but is omitted if it does not exist. The semantic devices below are used to show these differences. The properties apply to attributes within an entity and relationships alike.

M = Mandatory
O = Optional
MA = Mandatory if Applicable

The key for the Bibliographic Model (BM) (usually shown by an asterisk or is underlined), is the BRN. The designate entities in the ISBD also adopts the same key value for the BRN. In a relational database with Bib relation (having attributes BRN, title, Author) and another publisher relation (having attributes Brn, Place, Publisher and Date), the full record is represented by a join of the two relations.
4.9 Entity Analysis and Normalisation of AACR2R Elements

The different elements as outlined in the second level ISBD format (Fig 4.8 above) are each analysed in turn, area by area.

4.9.1 Title and Statement of Responsibility Area

The title and statement of responsibility area comprise of the following elements:

a. Title proper
b. General Material Designation
c. Parallel titles
d. Other title information
e. First statement of responsibility and each subsequent statement of responsibility

The Title identifies a document, summarises its content and assists in the filing and sorting of a record among other records (Rowley, 1989:10). Individual manuscripts may have a title, manuscript collections generally lack a formal title and therefore the cataloguer must supply a suitable title by examining the manuscript collection
which serves as chief source of information or by using a suitable reference source, published sources or other, for example, finding aids and inventories (Rule 4.0B).

Identification of the various elements within the area can be problematic. Tucker (1990:242) comments adding that 'the matrix of title information has two definite sub-categories of 'title proper' and 'other title information'. There is not always a definite 'mark' to distinguish the title proper from the other title information, for example, the title 'The province of Agra its history and administration' has no distinguishing marks but can nevertheless be translated into 'The province of Agra: its history and administration' (Tucker, 1990:241).

The parallel title is 'The title proper in another language and/or script' (AACR2R, 1988:620). This is not to be confused with the alternate title which is part of the title proper, for example, Marcel Marceau, ou, L'art du mime' (AACR2R, 1988:18). The parallel title is usually preceded by an equal sign. A description may have more than one parallel title at the second level of
description, (Rule 1.1D2). The parallel title itself can have other title information (ISBD(G), 1977:7), but is not 'independent' of the title proper.

**General Material Designation (GMD)**

The General Material Designation (GMD) is 'a term indicating the broad class of material to which an item belongs' (AACR2R, 1988:618). For manuscripts, the GMD of 'manuscripts' applies. For manuscript in microfilm the term 'microfilm' is used as the GMD. Specific forms, for example ms for handwritten manuscripts, typescripts, holographs are mentioned in the note area (AACR2R, 1988:133). In a mixed catalogue the GMD'S (see Appendix A for a full list), for example, manuscript, microfilm, machine-readable data files, are appropriate (Hensen, 1983:15). In the study only manuscripts are investigated. Further the element is optional. Hence the GMD is omitted from the description, but is nevertheless included in the model.

**Statement of Responsibility**

The statement of responsibility identifies and characterise the document further stipulating the nature and role of those responsible for the work, for example, person, corporate bodies, person as editor etc. (Rowley,
1989:10). The element is usually identified by a backslash (/). However, the statement of responsibility is sometimes not distinguishable from the title and forms an integral part of the title, for example, Marlowe’s plays, in which case it is transcribed without the use of a ‘/’ (Rule 1.1B2).

While the normal order of, title proper followed by, the GMD, parallel title, other title information and statements of responsibility, reflect the normal business rules, these are violated in the case of collected works without a collective title or in the case of parallel titles. In terms of Rule 1.1F11 the parallel title appears after the statement of responsibility and other title of information if an item has parallel titles and a statement or statements of responsibility in more than one language or script, thereby upsetting the normal order of the statement of responsibility as last element in the area (ISBD(G), 1977:2), for example:


When an item has no collective title, the predominant or first title is used, the parts being separated by a ; (Rule 1.1G), for example:
Clock symphony (no. 101) ; Surprise symphony : (no. 94) [GMD] / Haydn. (AACR2R, 1988:29).

The titles themselves are not together when different statement of responsibility are present, for example:


In modelling the elements within the area, cognizance must be taken of the deviations of the standard order, and provisions made for such deviations accordingly (the subject of chapter 6 of this thesis). Within the area, the following elements are identified having the following properties:

- An entry must have at least one title, with one title proper (which includes alternative titles). The title proper is mandatory, and where there is no title one is defined or else one is constructed (Rule 1.1B7). The title may have zero, one or more other title information and is mandatory if applicable.

- The GMD is optional and occurs only once. It is not part of the title but applies to the document as a
whole. Hence it forms part of the ISBD entity.
- Parallel titles are mandatory if applicable and may occur zero, one or more times. Each parallel title has zero, one or more other title information.
- The Statement of Responsibility is mandatory if applicable and occurs zero, one or more times.

The entire area which is itself mandatory, is modelled as in Fig 4.10. The figure shows the dependencies of the elements within the area relative to the larger ISBD entity, both of which are identified by the BRN.

Fig 4.10 Title Area Elements (ISBD, level 2)

4.9.2 Edition Area
The edition identifies and characterise a document to give an idea of the standing of the document (Rowley, 1989:10). The edition states whether an item is the 2nd, 3rd or revised edition of a work and the person
responsible for the revision. The first edition is not mentioned. The edition and responsibility are both MA. The edition is the primary element in the area. For each edition there are zero, one or more statements of responsibility. However, a statement of responsibility cannot exist if no edition exists. The edition area is represented by the following entities as illustrated in Fig 4.11.

Fig 4.11 Edition Area Elements (ISBD, level 2)

4.9.3 Date Area
The Date Area consists of zero, one or two dates. The Publications, distribution area has been renamed the Date area, since only dates form the elements in this area. Publisher and Place of publication details are excluded, as the publisher and place are omitted for unpublished items.

The Date(s) indicate the currency of a work (Rowley, 1989:11). Dates are not included in the area if they are already included in the title (Rule 4.4B1). Hence the
elements in the area mandatory if applicable. Hensen (1983:13), proposes inclusion of dates as the last element in the title. The AACR2 rules, however, provides for greater flexibility than the APPM by allowing dates to appear in the title or the Date Area. The rules make provision for a single date or inclusive dates giving the year(s), and optionally, the month and day. The following are some examples: 1945; 1945-1989; 1813 Dec. 17; 1813 Dec. 17 - 1814 Jan. 3. Dates, are however, problematic in that the format is not always standard. The following are possible variations for the date as indicated by Hensen (1983:13):

[1892?] - Probable Date
[ca. 1892] - Approximate date
[not before 1875] - Terminal date
[not after 1916 July 16] - Terminal date
[1814 or 1815] - one year or the other
[between 1906 and 1913] - dates less than 20 years apart
[189-] - Decade certain
[189-?] - Decade uncertain
[18--] - Century certain
[18--?] - Century uncertain
The above variations suggest the potential problem of dates when a strict format for dates, for example, '1945' or '1983 Dec 12' is used. This is particularly evident in the following examples '189-' or '1814 or 1815'. The Dbase date format mm/dd/yy cannot be used to represent dates and this poses an obvious design problem. The fields, date1 and date2 are, however, used tentatively to represent the opening and closing dates. The elements in the Date area are mandatory if applicable with zero, one or two dates appearing per description. Date 2 cannot exits without the existence of Date 1. The elements in the area are illustrated in Fig 4.12.

Fig 4.12 Date Area Elements (ISBD, Level 2)

4.9.4 Physical Description Area (PDA)
The data in the area records the physical appearance of the document including the number of pages, leaves, plates and height. The area comprises of three basic elements namely, the extent of item, other details and dimension. The dimension is not normally given for
manuscripts (Hensen, 1983:19). When given it must follow the other physical details. The following are some examples:

<table>
<thead>
<tr>
<th>EXTENT</th>
<th>OTHER</th>
<th>DIMENSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>[20] leaves</td>
<td>vellum, ill., maps</td>
<td>20 x 30cm</td>
</tr>
<tr>
<td>5 boxes [ca. 100 items]</td>
<td>parchment</td>
<td>28cm. or smaller</td>
</tr>
</tbody>
</table>

(AACR2R, 1988:130-132)

The information in the area is useful in determining the nature and uniqueness of the item. Sometimes the pagination may be decisive in determining the edition or printing of the work by examining the plates and pages. The illustration statement provides information concerning the way the work is presented (Rowley, 1989:11). The extent of the item is mandatory, while the other details and dimension are mandatory if applicable. The elements in the area are modelled as follows:

```
[M]
BRN PHYSDESC
```

```
EXTENT DIM OTHER
[M] [MA] [MA]
```

Fig 4.13 RDA Elements (ISBD, Level 2)
4.9.5 Notes Area

The notes contain useful descriptive information that cannot be fitted into other areas of the description (Rowley, 1989:11). The AACR2R provides for a variety of notes relevant to manuscripts. This includes inter alia, a summary, notes on the form and content of the collection, a thesis note, notes on published version of the manuscript, on the donor, physical description, notes on the language, title and any access restrictions (Hensen, 1983:26).

The summary may record any information on the creator of the manuscript, which may be a person, family or corporate body. A biography may include the place of birth, domicile, occupations, maiden names, pseudonyms, significant accomplishments etc. For corporate bodies, this may include information on the functions, purpose, and history of the body, its administrative hierarchy and previous, successive or variant names (Hensen, 1983:22).

Notes are mandatory if applicable, with zero, one or more notes given per entry. When designing a system, each type of note should be identified uniquely, for example, when printing a list of theses, a thesis note from the
database must be extracted from all other notes, for inclusion in a list. Hence an indicator for the type of note (Note-ind) and a field indicating the description (Note-desc) are used. The notes in the area are modelled as follows:

Fig 4.14 Note Area Elements (ISBD, Level 2)

4.9.6 Main Entry Heading

Traditionally, one of the tasks of cataloguing rules is to determine which of the several claimants to a principal position in a bibliographic record is to be considered as the main entry (Carpenter, 1989:88). Manuscript records are usually formed around a person, family or corporate body. These are usually chosen as the main entry or chief access point, that is, the entity responsible for the creation of the work becomes the main entry (Hensen, 1983:5). The main entry heading can be any one of the following, a name, corporate body, title or uniform title. The uniform titles collate the different editions and translations under a common title. Entry
under their own titles would impair the systematic character of the catalogue (O’neill & Vizine-Goetz, 1989:168). The Main Entry heading is optional in an alternate entry system, otherwise, it is mandatory. In a very general sense the main entry heading is mandatory if applicable and this depends on whether the alternate entry or main entry approach is used. The relationship between the typical bibliographic entry, called the Bibliographic Model (BM) and the main entry is illustrated as in Fig 4.15. Figure 4.16 illustrates the total entity model resulting from the culmination of the individual entity diagrams developed thus far.

Fig 4.15 Model showing relationship of the
Main Entry Heading in the BM

202
Fig 4.16 Entity Model (BM) for Manuscripts, Level 2
Even while manuscripts were recognised as distinct from non-book material, the framers of the rules realised, however, that, the rules as outlined in chapter 4, would not be sufficient for use in archives and manuscript libraries.

'The Rules are designed for use in the construction of catalogues and other lists in general libraries of all sizes. They are not specifically intended for specialist and archival libraries, but such libraries are recommended to use the rules as the basis for their cataloguing and to augment their provisions as necessary (AACR2R, 1988:1)'.

4.10 Criticism of AACR2R: Manuscripts
The AACR2R shows a decided lack of understanding and appreciation of how collections are arranged, described and controlled⁷. For instance, it recognises the need for item and collection level cataloguing but fails to acknowledge multi-level arrangement and description for archives and manuscripts. A collection can have up to eight or nine levels. Hensen’s APPM (Archives, Personal papers, and manuscripts: a cataloging manual for archival repositories, historical societies, and manuscript libraries) is the best critique on the AACR2⁸ and raises many points of inadequacies in the AACR2 code for the cataloguing of manuscripts. The APPM uses the
AACR2 as a point of departure in developing a set of rules consistent with the cataloguing, description, and arrangement of manuscripts and archives. In doing so common aspects of descriptive elements of both professions are acknowledged, but with significant differences. The APPM sets to correct the deficiencies in the AACR2 and features of archival description and arrangement such as archival series and record groups, which the AACR2 had failed to address, are included (Weber, 1988:505). Even though Hensen criticises the AACR2; moreover, he admits that 'with appropriate modifications, library-based descriptive techniques can be applied' to archives and manuscripts (Hensen, 1983:1). The following are some of the points raised.

In the AACR2 individual manuscripts, collections and archives are treated similarly in terms of description. According to Hensen (1988:542) the fundamental flaw with the AACR2 rules is that it does not cater for manuscripts and archives based on archival traditions and practices but instead adopts a book-oriented approach to cataloguing. Too much emphasis is given to the cataloguing of individual items and little attention paid to the cataloguing of large collections. Hensen elaborates further stating that 'The size of most modern
manuscript and archival collections has led to archivists and manuscript curators to prefer limited cataloging control over all of their holdings rather than detailed control over only some of their holdings, thus, collective description is the only practical response to the overwhelming cataloging burdens that would be presented by item level description’ (Hensen, 1983:2). Treating collections as a unit, the entire whole is seen as the sum of the interrelationships of its parts. The individual item derives its importance from its context within the collection. In most manuscript cataloguing, the analog to the book is not the individual manuscript, but the collection of which the item is a part. Adopting the collection approach, on the other hand does not exclude single item ‘collections’ which have no particular collective context and must be catalogued strictly in terms of their own format and content (Hensen, 1983:2). The AACR2 has failed to see the role of collections as being more prominent than individual manuscripts.

The AACR2 failed to address critical issues of arrangement like ‘archival record groups and series’ for which different descriptive techniques are adopted at different levels. There is no uniformity of descriptions
as in the case of books. Concerning the arrangement of manuscripts and archives, Hensen (1983:2) asserts that 'It is recognized as fundamental to archival and manuscript cataloging that there are several appropriate levels of description. Most of these levels are determined by natural groupings based on the provenance of the materials. Nevertheless there are sound reasons for cataloguing manuscripts and archives at less than the largest logical level. ... it will occasionally be necessary to make separate catalog entries by way of calling special attention to these items'. The AACR2R makes provision for item and collection level cataloguing at only two levels (analytics chapter 13) but fails to make provision for three level description (analytics within analytics) as manuscripts do not have a series area in which the third level is recorded.

Central to the whole philosophy of manuscript and archival cataloguing is the role of the finding aid, guide, or series description prepared during the course of processing and analysing a manuscript collection. Different kinds of finding aids may exist at different levels. These may include descriptive registers, inventories, calendars, indexes, shelf and container lists. The catalogue description often serves as a
repository index to these finding aids. It is against the standard practice of the total archival management process that the catalog must be placed within context.

The AACR2 and AACR2R have not recognised the prominent role of the finding aid as the chief source, even though the precedence outlined in the AACR2R give the following order; another manuscript copy of the item, a published edition of the item, reference sources and other sources. The chief source for a manuscript text is the manuscript itself and for collections, the collection itself (AACR2R, 1988:123). However, the rules blunder in that emphasis is still placed on preference for information taken from the title page, the colophon, caption headings and lastly the text itself. The finding aid has no place in the priorities listed above. In actual practice these finding aids are approached by users as a surrogate for the whole collection and are in effect, the only practical equivalent to a chief source of information. They are created during the course of the collection’s processing and give reliable information on elements such as titles, dates and physical extent just as concretely as does the title page of a book (Hensen, 1983:3). From an archival point of view the finding aids function as the chief sources of information.
Hensen identifies, difference in practices concerning the various elements in the ISBD areas. With respect to the title, for instance, the title consists of three subfields, the name or names of one or more persons associated with or responsible for the item or collection; a statement for the form of the material (letters, diaries, correspondence etc.) and the date or inclusive dates of the item or collection (Hensen, 1983:11). The AACR2 does not recognise this structure, but does not specify that the date must be in the title area. Instead provision is made for the date to be in the title or the date area and the form to be in the title or in a note.

Hensen identifies some important notes omitted in AACR2, for example, a note reflecting the 'relationship complexity' of the item or collection to a larger unit or whether it is part of another unit. The relationships are vital in identifying the various units or parts of a collection that may complement each other or are catalogued separately. For instance institutions tend to microfilm or photocopy documents owned by individuals or other institutions. In such cases it is desirable to make a note of its origins, for example, 'Microfilm of
original in: Yale University Library (New Haven, Conn.)' or 'Photocopies (negative) of originals in: Archives nationales (Paris, France)' (Hensen, 1983:21). Similarly, a note identifying any finding aids linked to the entry, for example, inventories, calendars, card catalogue, indexes, guides etc. are useful to the user (Hensen, 1983:27).

With regards to the form of the main entry heading, the principle adopted for entering personal names is under the 'name by which he or she is commonly known' (Rule 22.1A). From an archival point of view, it is more important that while Samuel Clemens is more popularly known as 'Mark Twain', that he 'lived his life and generated his papers under his given name'. Similarly, corporate bodies entered 'directly under the name by which it is commonly identified' (Rule 24.1) fails to show the full administrative hierarchy in a heading (Hensen, 1983:6). The statement of responsibility for manuscripts on the other hand, relate more to provenance than to the creative responsibility existing between the author of a book and the book itself (Hensen, 1983:3). The use of the statement of responsibility would duplicate the main entry. Ideally when the title is
created the responsibility statement is integrated into the title and forms part of the title. Its use as a separate element in the description is questionable.

The attempt by the framers of the AACR2R code to treat manuscripts similar to books has created differences in the way manuscripts are catalogued, described and arranged. These differences are explored in the next chapter and comparisons made concerning the principles, techniques and practices of both the professions. This is followed by an in-depth investigation of the NAREM requirements.

NOTES

1. Information taken from Library of Congress cataloguing in Publication Data. Hence no page number is available.
2. Discussed in section 4.6 of this thesis.
3. If different from main entry heading in form or number or if there is no main entry heading.
4. The example has been amended to conform to the ISBD format.
5. See Fig 4.11 for a normalised diagram for the edition.
6. The latter is an important consideration in the
implementation design phase (discussed in chapter 6 of this thesis) when storage issues and denormalisation is considered to arrive at an efficient physical design. The edition for instance, can be denormalised, as it is a designate entity.

7. Criticism of the AACR2R is limited to discussion of manuscripts only. The reader interested in other aspects may consult Simonton (1979), Vervliet (1976), Ayres (1980) and Verona (1980).

8. The criticism for the AACR2 also applies to the AACR2R as the revised edition does not have significant revisions.
CHAPTER 5

NATIONAL REGISTER OF MANUSCRIPTS

5.1 Introduction

The archival tradition is based on the principle of provenance which asserts that records of different individuals and organisations have an organic relationship to those individuals or organisations and should therefore be kept separate and never mixed (Holmes, 1964:25). The archives reflect an organised activity, the records themselves having been collected during the lifetime of the individuals or organisations to which they belong.

The principle of provenance was given theoretical justification in a Dutch manual published in 1898. Before then (and even much later) records were regarded as discrete items and were catalogued and classified according to library techniques (Hesselager, 1984:263). Gertz & Stout (1989:7), on the other hand, state that as recently as forty or fifty years ago archival and manuscript material were routinely catalogued as discrete items. The overwhelming bulk of archival material saw a departure from this method and gave way to group
descriptions rather than catalogue each letter or memo, for instance, to enable quick organisation and description of archival material. Instead bibliographic control placed emphasis on collection level descriptions. The result was the development of a tradition with its own set of principles and practices, termed the archival approach. Hence, there was a need in this study to look more closely at the kinds of differences between archival and library practices in order to gain an understanding of the processes, principles and techniques involved. However, first a few formal definitions of manuscripts and archives are given and this is followed by a discussion on the differences between archives and manuscripts.

5.2 Archives and Manuscripts

Definition of Archives:

'The preserved documentary records of any corporate body, governmental agency or office, or organization or group which are the direct result of administrative or organizational activity of the originating body and which are maintained according to their original provenance' (Hensen, 1983:8).

Archives are 'media which had been generated by an organisation in the course of its business, and which had turned out to be worth keeping' (Cook, 1986:5). Unlike manuscripts which look similar to archive records,
archives are usually administered by the organisation that created the records, by the parent organisation or delegated to another institution or service. According to Cook (1986: 5), collecting manuscripts is not the same as exercising the duty of archival management.

Cook (1986:7-8) draws the following distinction between archives and manuscripts. The material in an archive is drawn from usually it's employing body. A manuscript library acquires materials from a more or less arbitrary system of marketing. An archive collection may also contain items acquired originally from outside the organisation. Manuscripts bought by the library may exhibit archival characteristics and may even be the archives of an organisation which had disposed them off. In this case the library exercises an implicit delegated function. The materials may differ less than the method by which they were acquired.

An American archivist, T R Schellenberg, contrasts manuscripts and archives in terms of their origin and organisation stating that 'While archives grow out of some regular functional activity, historical manuscripts, in contrast, are usually the product of a spontaneous expression of thought or feeling. They are thus
ordinarily created in a haphazard, and not in a systematic manner' (Schellenberg, 1956:18). In archival collections, the original order is usually satisfactory and is mostly retained because archives are usually the product of a records management program (Berner, 1965:214). The original order may not be clear in which case a decision to retain the original order or to impose a new one, is made.

Do these similarities and differences between archives and manuscripts mean that manuscripts and archives (MAR) have to be arranged and described differently? While the two traditions exist, Cook (1986:6) concludes that 'it is not possible to separate them in a discussion of how to manage archival materials'. Schellenberg asserts that private records including personal records, have the same 'organic quality' as public records and this justifies their documentation according to archive rather than library techniques (Duminy, 1986:23). Hence many books on archives or manuscripts have invariably treated them as one unit.
5.3 Similarities and Differences: Books, Manuscripts and Archives

Evans (1986:250), remarks that 'archival management of records is based on the assumption that context is the key to understanding. Archival principles of provenance and the respect des fonds, the sanctity of original order, and the notions of hierarchy and the organic nature of archival materials all derive from this fundamental concept'. Archives and manuscripts differ from books not only by the basic principles on which they are based, but by the emphasis on description of collections instead of items. The different levels of arrangement and many types of finding aids used, mark some of the differences between that of manuscripts and books. These areas are explored.

5.3.1 Provenance

Archives are not defined by their physical medium, like video-recordings, books, maps, or by the condition of their issuance like serials, but rather by the principles of provenance and original order by which they are described and physically organised (Fox, 1990:19). Because collections grow organically from the activities, lives and actions of organisations and

217
individuals, the archivists must respect and preserve such provenance. Provenance provides the context in which the records become intelligible and is intrinsically bound with the life of the individual or functions of the organisation (Fox, 1990:20).

Provenance is defined as 'place of origin' (Concise Oxford Dictionary, 1982:829). However, in terms of records management, the organic nature, the origin and the interrelationship of the parts to the whole are of primary significance in defining provenance. Hensen (1983:2) defines this principle 'The principle of archival unity, which recognizes that, in organically-generated collections at least, it is the collective whole as sum of the interrelationships of its components that has significance and that individual items or sub-series within a collection usually derives its importance from its context'. Holmes (1964:25) states that 'The basic principle of respect des fonds requires that the records of different creating agencies and offices be kept separate and never mixed'. The essence of this rule, is that records normally created in relation to a specific function or work reflect that activity and
consequently should be kept together. They should not be merged with records created by other persons or organisations even if their subject content is the same.

By maintaining the integrity of the records according to the source that generated them, the user will be able to analyse the actual work of the person or organisation in the truest context, that in which the work transpired (Berner, 1965:213). According to Blouin (1986:158) original order is as important as the information contained in them. Original order preserves context which often enhances the content of an archive. To impose an artificial arrangement scheme would jeopardise the papers validity as historical evidence (Gertz & Stout, 1989:10).

5.3.2 Unit of Description
Collections rather than individual items are the primary units of intellectual analysis. Sometimes an entire collection might consist of a single diary, letter, minute book which are described individually (Fox, 1990:20). A collection consists of materials that have come together in the course of some human activity, for example, the documents created or accumulated by Alan Paton as author, politician and humanitarian may become the Alan Paton Papers. Similarly, for government
departments, the analogue to a collection is the record group, comprising of all documentation created by that organisation to meet its managerial and custodial responsibilities. In both instances, it is the common provenance that binds the material together into a coherent identifiable unit (Fox, 1990:19-20). Even though the material is varied and in some cases the description and identification of individual items is justified, the individual item gains its importance in relation to the other items.

5.3.3 Arrangement
Unlike published works which presents the cataloguer with a fixed internal structure, that is, each book is organised into chapters following a decided sequence, the archivist is frequently faced with the necessity of determining a proper internal order and arrangement for a collection. The guiding principle here is respect for and retention of the original order since this reflects the way the records were created and used (Fox, 1990:21).

When archives and manuscripts are arranged, the physical sort reflects the original order of the system by which the documents were originally created and used. This involves arrangement rather than classification.
Arrangement implies ordering the documents in relation to one another and not into an order which is a pre-established analysis of concepts or subjects (Cook, 1986:33), as is the case with traditional classification. Archival arrangement does not usually involve the use of classification schemes since the material cannot be sorted into predetermined categories. Where general classification schemes have been developed for archives, the classification provides a system of reference codes which reflect the relationship between the components of a collection, the main function of the classification scheme is to formalise the archival order established during the process of arrangement and relate it to administrative control instruments such as indexes and other finding aids (Cook, 1986:90-92).

5.3.3.1 Levels of Description and Arrangement
Material in a collection are arranged in hierarchies, the highest level is the 'record group' followed by sub-groups representing lower levels in a hierarchy, for example, divisions of an organisation. The next level is the series, which is not to be confused with the librarians use of the term, that is, separate items sharing a collective title. The archival series is an accumulation of records having similar characteristics
such as the document format, filing order, or subject matter. Below that level the individual file folders, volumes or other filing units or items are arranged (Gertz & Stout, 1989:10).

No set number of levels are fixed for archival descriptions. Cook (1980:22), for instance, identifies 5 levels of description for archival material; a group, sub-group, series, sub-series and unit levels. The group level is the major division of archival holdings based upon some common provenance; the sub-group level is an organic subdivision of the group. The series or class level is the basic division of archives into cohesive describable units, usually large but capable of control as single items. The group, sub-group and series levels (levels 1-3) are macro-level descriptions and usually consist of a set of summary descriptions. Levels 4 and 5 (subseries and unit levels) are micro-level descriptions. The sub-series level is a division of the series based upon some common feature with the series but describable as an independent unit, for example, a set of reports of a division. The unit level is the basic physical unit consisting of individual documents or files.

222
Cook (1980:22), asserts that there is a general assumption that archives have at least two levels. The primary components consist of a set of summary descriptions at a fairly major level (macro-description). The secondary component consists of micro-descriptions. A particular user may go first to the guide (descriptions at levels 1-3), and this will direct him to the more detailed lists covering the series he is interested in. Elsewhere, Cook (1986:44) states that descriptions are based on two levels at least, a description of the class and a finding aid which gives access to the individual documents forming part of it. However, in a small collection, perhaps in a local history collection of a library, a single level of description usually a macro-description will allow sufficient control (Cook, 1980:22).

Of particular interest to this study is the 5 levels distinguished by Holmes (1964). He not only identifies 5 different levels but mentions different finding aids used at each of the different levels specified. ‘The term ‘arrangement’ covers several different types of operations, of varying degrees of difficulty, depending in a large part on the level at which they are being carried out’ (Holmes, 1964:23). The 5 levels of
arrangement identified by Holmes are, the depository, record group/subgroup level, series, filing unit and document levels.

A. Depository level

A depository may be broken down into a few major divisions which usually relate to the physical placement of the holdings and administrative responsibility for the major groupings within the repository. The basis of such arrangement may be chronology (for example, colonial papers and papers belonging to the Union Government); hierarchy (for example, where each unit is successively subdivided into smaller units); or based on levels of government, for example, central and local government. At the level of the depository, the organisation and arrangement may be accompanied by a guide to the broad subject categories in the archives (showing what's where). Important matters such as the size and arrangement of the storage areas, the physical nature of the records themselves as in the case of maps, pictures, microfilms etc. that require special storage means and conditions, for example, security, are noted in the guide (Holmes, 1964: 24-25).
B. Record Group and Subgroup Level

This level is represented by a breakdown of the holdings of the administrative division or branch (level one) into record groups and their placement in the physical area according to some logical pattern in the stack areas. The groups themselves can be ordered and broken down into subgroups (Holmes, 1964:24).

Based on the principle of provenance (respect des fonds), records of different creating agencies and offices are to be kept separate and not mixed. The creating agencies or record/archive groups thus formed, are usually the basic unit for administrative control; that is for arrangement, description, reference service, statistical accounting and reporting. Each paper, document or report must be identified with only one record group. Records cannot be physically placed in two different places. Each accession or collection must be allocated to its proper record group. Any subgroups must be determined and their affiliations determined when the boundaries of the record group are established (Holmes, 1964:25-28). Relatively speaking, the arrangement at the record group level occupies little time as
compared to the time taken to arrange the collection into series, filing units and documents.

C. Series Level
The record group itself is broken down into natural series. Arrangement at the series (and lower levels) represents time-consuming work, some record groups extending to many linear metres containing thousand of series. The arrangement by provenance into series forms the heart of archival work, thereafter the series itself is ordered logically in relationship with each other. Therefore knowledge of the administrative history of the agency, change in functions and organisation, history of the preceding agencies of which it is part or successor, is useful in interpreting the many series and accomplishments of the organisation (Holmes, 1964:28-29).

A true series is composed of similar filing units arranged in a consistent pattern within which each of the filing units has it's proper place. A series has a beginning and an ending with appropriate sub-divisions by alphabetic, numerical, chronological, country, state etc.; the boundaries of a series is more or less well defined, for example, the result of a special operation
or event. Occasionally, a series may consist of only one file unit or volume. It is the series that form the inventories which at this stage exists as information on slips of paper or cards (Holmes, 1964:30-31). A catalogue description is usually prepared for the series and a unique accession number is assigned to each series (Gilmore, 1988:613).

D. Filing Unit Level

Each series, in turn, is broken down into one or more filing units. Each component is arranged in some logical sequence, a sequence which usually has already been established by the agency so that the archivist merely verifies and accepts it. Filing units may be single documents or groups of documents relating to some transaction, person, case or subject. Often such filing units are folders or a group of folders representing a class of documents (Holmes, 1964:33). The files are arranged, labelled and placed in boxes in alphabetical, numerical or subject order.
E. Document Level

Individual documents within each filing unit are arranged together with any enclosures, annexes and individual pieces of paper that comprise the filing unit. The documents are placed in relation to other documents in some accepted and consistent order (Holmes, 1964:24).

Once the collection is analysed and arranged, the items are boxed, using the file units and series to function as breaks in the boxing. Each box should contain the record group (name and number), any subgroup identifier, the beginning and end file numbers. A sequence of numbers should be established for boxes within the record group, subgroup or series thereby giving it 'uniqueness'.

Berner (1965) forms a broad theoretical framework for a common approach towards the arrangement and descriptions for manuscripts. In essence this does not differ from that of archival treatment. The 'Record Group' among archives is equivalent to the 'Manuscript Group' in a manuscript collection. Both record and manuscript groups are composed of series which are composed of file folder units, which in turn are comprised of items. The arrangement and description can be keyed to any of these
record levels depending on the kind of detail desired. For instance, in establishing file-folder control, the folder heading, a brief description of the folders content, including dates and names of significant correspondents can be given. One way of control at the file folder level is to number each file folder serially to give quick access to the files for retrieval (Berner, 1965:218). Each item may be described individually, depending on the need to describe the items at that level.

5.3.4 Finding Aids: Catalogues, Guides and Inventories
Even though the collection can be identified at the collection level by a catalogue card, for instance, the sheer bulk of the components within them, the variety of forms of material, their varying subject content, justifies different techniques and finding tools to reveal aspects of a collection. The catalogue is an important component in the system of descriptive finding aids but certainly not the only one used in manuscript and archival institutions. The catalogue serve as pointers to other descriptive tools like registers, inventories, indexes, lists, calendars and directories (Fox, 1990:21). Sometimes this may be due to the level of description, while in other cases, on the nature of the
material itself, for example, a name index for a collection may be constructed and added to an existing cumulative name index.

Finding aids not only help users to find and locate information they require but form a means of administrative control whereby staff can ensure that the items are properly accessioned, processed, stored and retrieved. It allows staff to ascertain the location (shelf-number) and physical characteristics of the collection (for example, size and condition) (Cook, 1980:23).

Researchers usually first consult the guides to the archives holdings and then selected finding aids for individual collections to determine which boxes are relevant to a query (Gertz & Stout, 1989:7). Just as a catalogue card serves as a surrogate for the entire book, guides provide an overall description for the collection as a whole⁴. Details included in a guide entry are, inter alia, the title, beginning and ending dates of a collection, a summary and associated keywords. The guide is a summary of the contents of the entire collection with bibliographic type elements such as authors, titles and subjects. Of all the elements, the
summary is the key to the contents of a collection. Invariably the summary contains categories of information (for example, newspapers, correspondence, reports, miscellaneous, speeches etc.), thereby providing an overview of the content of a collection.

The inventory is the finding aid mostly employed in the description of records. In practice when an accession/donation is received, it is entered in an accession register. An accession number is allocated, the terminal dates, a summary, description of the collection, the name and address of the donor and the correspondence file reference are noted. Lastly, any applicable explanatory remarks are noted down, for example, accessibility restrictions (Olivier, 1980:38). Once the items are sorted, a data form is prepared and an index or catalogue card is prepared for the benefit of the researchers. Should a donation consist of a single or only few documents, the descriptive note in the register may be sufficient. If the donation is very large, it must be properly sorted and described in an inventory (NAREM, 1985:2).
The inventory is defined as 'a basic archival finding aid that generally includes a brief history of the organization and functions of the agency whose records are being described; a descriptive list of each record series giving a minimum such data as title, inclusive dates, quantity, arrangement, relationships to other series, and description of significant subject content; and, if appropriate, appendices which provide such supplementary information as a glossary of abbreviations and special terms, lists of folder headings on special selective indexes' (Buys, 1982:2-3). Appendix B has an example of an inventory. The inventory is more than just a finding aid but reflects the arrangement of a collection. It stands also as the most thorough single description of an entire collection for it combines data on both arrangement and content. The inventories are not designed to present the researcher with a sense of all potential uses of the material or subjects covered but it provides the researcher with a detailed sense of the relationship of the existing record to the activity that generated the record. Using the inventory the researcher should be able to locate relevant material (Blouin, 1986:161). An inventory may, in addition, serve as a pointer to lists, calendars and indexes that provide greater details on specific aspects of a collection.
Each level of manuscript and archival control tends to require a different type of finding aid. In terms of the levels identified by Holmes, reports or finding aids can be generated at the depository level, showing administrative aspects. At the record group level details of the record group are given. At the series level an inventory is constructed. At the file unit level, a checklist of filing units for important series is provided. At the document level, a written statement is not normally produced to reflect arrangement (Holmes, 1964:40-41). Catalogue descriptions can be prepared for any of the levels, ranging from control at the depository, series, file folder or a single piece therein.

The term archival description is preferred to cataloguing because more than one description is prepared for the different levels of archival description and is not limited to a description for the catalogue only. The variety of archival formats make it difficult to establish only one method of finding and retrieving material. Hence a complex system of finding aids are used to exploit the information in a collection, unlike the catalogue which has a more or less fixed format for
describing materials that can generate its own set of problems. Cataloguing at the item level, for instance, can lead to an abnormal clutter of the catalogue.

5.3.5 Stability and Change
Library materials are catalogued once, the author will generally not change over time (unless there are rule changes in the cataloguing code). Details like the title, dates, subject headings, extent of item are stable. New editions have new records created for them. Manuscript and archival records on the other hand, are much more dynamic. Family members of an existing donor may contribute additional parts to a collection. Under such circumstances, the title, dates and content notes may have to be amended to reflect the changes in the new state of the collection (Roe, 1990:152).

5.3.6 Format
A collection might contain items in only one format or many different formats. It might consist of photographs, maps, textual material, audio-visual material or combine a mixture of one or more categories (Fox, 1990:21). Since collections accrue in the course of a person’s or organisation’s activity, they are not necessarily media specific but are more often than not, of a mixed media.

234
5.3.7 Uniqueness
Library materials are generally mass produced in multiple copies. These include different editions and translations of the same work (Roe, 1990:148). Each collection on the other hand is unique. While certain pieces within the collection may exist in multiple copies, the collection as a whole or parts thereof are, however, unique (Fox, 1990:21), for example, a group of letters to and from a person is a unique entity.

5.3.8 Descriptive Elements
Most collections lack the formal elements found in publications. The titles, statements of responsibility and publisher details for published material are usually found in title pages, reference works or any published and established source from which the details can be extracted. A description or title for instance that is constructed for a collection, when formulated, is not necessarily universally understood as in the case of books. Hence, a title given to a collection is often unique and is not familiar to the average user. As such, the descriptions, while appearing in a similar format and structure as bibliographic record descriptions, are more
explanations for the user in an attempt to convey the nature, scope and content of the collection (Fox, 1990:21).

The nature of the elements vary in both practices. The title is formulated rather than transcribed. The title both describes and uniquely identifies the collection. Collections span a period of time, the inclusive dates of the material are given. The dates may however be misleading, as the bulk of the collection may fall within a narrower time frame and the bulk dates, for example, ‘Papers, 1875-1965 (bulk 1890-1927)’ are given to bring to the reader’s attention the key period covered (Fox, 1990:23-26). Concerning the physical description, there is a great variation in size and composition of the material including a variety of physical media and packaging. In cases more than one statement of extent is required, particularly when there are multiple physical media, for example, text and computer files (Fox, 1990:31-32).

5.3.8.1 Notes

The notes form an important component in a manuscript entry. The notes reflect the origin, content and structure of a collection. According to Fox (1990:26),

236
they are the heart of the record providing a means to the reader of the nature of the collection. Key notes mentioned by Fox (1990:26-33) are:

- Biographical/historical notes: The biographical or historical details place the material in the proper context of the individual or the corporate body.

- Scope and Content/Abstract Note: The content or abstract provides a narrative description of the general content, nature and scope of the collection. Included in the description are the various forms of materials found in the collection (diaries, maps, journals, case files etc.), important persons, functions, events and activities represented therein.

- Linking Entry Complexity Note: A linking entry note places the component part in hierarchical context to a higher or next larger unit in the collection. For instance the 'Mayoral Subject Files' that form part of the Humphrey Papers is given in a note. When the catalogue record describes a group of materials, such a series within a collection, the reference begins with the
phrase 'Forms part of', for example, 'Forms part of: Humphrey, Hubert H (Hubert Horatio), 1911-1978. Papers, 1919-1978'. If the record is for an individual item, an 'in' analytic is used, for example, 'In: Holman Municipal Airport. Records, 1927-1945'.

- Microfilm and Other Reproductions: Included in the description are notes on microfilming and other reproductions. The notes include details on the status of the microfilm in relation to the collection, for example, part of collection on microfilm. The notes may state that the original is held by another institution or that the originals have been destroyed after microfilming.

- Organisation and Arrangement: The note describes the manner in which smaller units are organised (for example, alphabetical, chronological, numerical, by subject, state etc.). The data is essential, for example, when land deeds are arranged by parcel number. It cannot be located unless the number is known. The details may reflect more than one method of arrangement, for example, case files arranged chronologically by date of admission, correspondence files arranged alphabetically or subject files ordered by subject.

238
- Provenance and Source of Acquisition: This note provides relevant details on the history and custody of the materials prior to the time of their acquisition (for example, previous ownership of another institution). Information about the donor or collector, details on the method of acquisition (donation, purchase, transfer), price and accession numbers are listed as well.

- Restriction, Access and terms governing Use: The notes describe various conditions governing the use and access to manuscripts and archives. Permission to use the material might be controlled by statute, available after a specific time period as specified by the condition of donation or require advanced notification to retrieve the material from a remote storage location. In other cases copyright restrictions or restrictions due to institutional policies (for example, reproduction of fragile materials), may hold.

- Cumulative Index/Finding Aids: In addition to subject heading access to the catalogue, additional finding aids such as registers, inventories, indexes, calendars, box lists that describe the collection or portions of the collection are noted. The note announces the existence of such finding aids, their type and availability.
Calendars, for instance, record an item by item description of correspondence, organised, chronologically.

Theodore Schellenberg, one of the most influential American archival theorists wrote of 'basic methodological differences between the archival and library profession', states that 'the materials received by a librarian are referred to as acquisitions, denoting purchases, gifts, and exchanges, while those of an archivist are called accessions, which are received by transfer or deposit; the librarian selects his materials, while the archivist appraises his; the librarian classifies his materials in accordance with established classification schemes, while the archivist arranges his in relation to organic structure and function; the librarian catalogs his materials, while the archivist describes his in guides, inventories, and lists' (Schellenberg, 1956:23-24).

Archivists are reluctant to describe the processing of archival and manuscript materials in library terms. Instead of cataloguing they prefer to use the term description. Arrangement is used in place of classification. Archival description is not a process of
bibliographically distinguishing a work from other similar works and archival arrangement has nothing to do with subject classification schemes (Gertz & Stout, 1989:6). As demonstrated above, the differences are significant yet in terms of descriptive elements used by both professions there are marked similarities and differences.

Smiraglia (1990) remarks that there are both similarities and differences between the two professions. Bibliographic material have bibliographic characteristics (for example, title, author, edition etc.) based on a tradition of bibliographic descriptive standards. Archival material are the raw stuff of creation, with no title page, chief source of information or statement of responsibility to rely on, except for formal finding aids to supply titles, dates, historical notes and other information for retrieval (Smiraglia, 1990:9). Unlike published material of which the user knows the title or author or some information on the work being sought, archival material are sought in ways that are more inferential, the archivist relying heavily on the reference interview, indexes, notes and nature of the collection itself to assist the researcher in locating the material (Smiraglia, 1990:10-11). However, the goals
of both the professions are to exploit recorded knowledge. The objective of bibliographic and archival tools are the same; only their priority and ranking differs.

5.4 Computerisation

The ability of computerised systems to produce registers, inventories, folder and box lists, catalogue cards and other kinds of finding aids, makes computerisation extremely attractive. Out of a single database a number of selected finding aids can be produced, for example, repository staff need shelf lists, conservation staff need a list of items requiring repair, searchroom staff need to produce guide entries, inventories, handbooks and indexes (Cook, 1980:16). In addition, computers are useful in controlling the various management processes like logging the items/collections received and processed, disposals and movement of materials between various depots. Further, computerisation can cater for all kinds of administrative information such as patron registration, accessioning, donor information and collection-use statistics (Weber, 1988:503).
Local developments for manuscript collections have been but a few, the most popular being the National Register for Manuscripts initiated by the Government Archives Services.

5.5 Government Archives Services

In order to put the administration of the archives of the Cape Colony on a sound footing, the government decided to appoint a parliamentary commission in 1876 to 'collect, examine, classify and index the Colonial Archives'. A report under the chairmanship of the Chief Justice, Sir J H de Villiers was submitted by the Commission and finally adopted on 13th January 1877. This document marked the beginnings of organised archival service in South Africa and the Cape Archives Depot in particular (Snyman, 1976:16).

By 1971 the holdings of the Archives Service increased to 51,000 linear metres (Ferreira, 1991:35). By then it was realised that new methods of capturing and retrieving documents was necessary to cope with the increasing backlog and increased volume in archivalia. A memorandum, dated 5 March 1971 by the Director of Archives to the Secretary for National Education, prompted a preliminary feasibility study on the introduction of computer
facilities in the Government Archives Services (GAS). The results indicated that the application of the computer had become imperative in the light of increased backlog, volume of the archivalia, and the need for sophisticated forms of information retrieval to satisfy user needs (Olivier, 1983:6). Computerisation would increase the control over the archivalia in the Archives Depots and Intermediate Depots throughout South Africa as well as meet administrative needs and the needs of researchers (Ferreira, 1983:2).

The new system began operation in 1974. The VIDEO/370 and ATMS III software are used for data capture and update and the Storage and Information Retrieval System (STAIRS) for retrieval (Ferreira, 1983:5). Today the service comprise of numerous databases. The databases include the holdings of, amongst others, the Transvaal Archives Depot; the Central Archives Depot (Pretoria); the Cape Archives Depot (Cape Town); the Natal Archives Depot (Pietermaritzburg) and the South West Africa Archives Depot. Each of the databases is identified by a unique 3 letter code, for example, TAB for the Transvaal Archives Depot, SAB for the Central Archives Depot and NAB for the Natal Archives Depot. Only data pertaining to the archives (that is, documents created or received by
government offices) are placed in these databases. The holdings of all the databases including the NAREM
database for manuscripts, maps and microfilms are combined into one database, the RSA database (Olivier,

A variety of services including finding aids (for example, lists, guides, inventories) and publications
(for example, statutes, parliamentary papers, debates) are published under the archives services. The
publications serve as a means of controlling the national resources. The List of Archivalia is by far the most
important, providing a point of departure for searching of archival material and forming the basis of the topmost
control of all finding aids followed by one or more specific guides for various categories, for example,
maps, photos, archives and manuscripts for each Depot or institution (Potgieter, 1982:19).

In 1978 manuscript collections in the possession of institutions other than the Government Archives Services
depots, such as libraries and documentation centres, have been included in the database as well. This project
became known as the National Register of Manuscripts (NAREM).
5.6 National Register of Manuscripts

The need to record and know what manuscripts are available in the country prompted the Bibliographic Committee of the South African Library Society in 1977 to request the South African Library to compile a national register for manuscripts. The project was brought to the attention of the Director of Archives by Dr J H Snyman (Chief of the Cape Archives Depot) and Tyrrell-Glynn, Assistant Director of the South African Library, the editor of the project. The Archives Services was already at this time engaged in the computerisation of its holdings. The Director of Archives proposed computerisation of the manuscript entries, to which Tyrrell-Glynn agreed. The project was approved finally by the Secretary for National Education on 25th November 1977 (Smith, 1978:27) and NAREM was born out of these efforts and began earnestly in 1978.

While the GAS concerns itself with official documents, it is in the national interest to make all documents accessible and available under the scheme for research purposes (Smith, 1978:29). The main objective is to record information concerning manuscript collections in all South African repositories at a central point, the

246
central point being the Government Archives and to make information available by way of guides and computerised retrieval (Olivier, 1980:34).

5.7 Components of a NAREM Entry and Implications for Design

NAREM is based on the idea of collectivity. Only complete collections are incorporated in the register. Each guide is comprised of many entries and each entry in a typical NAREM guide comprises of one or more paragraphs. Every paragraph begins on a new line. An entry is comprised of the following paragraphs:

1. Depot or repository
2. Type of Material
3. Reference Number
4. Title
5. Starting Date of Collection
6. Ending Date of Collection
7. Remarks
8. Summary
The order of the elements is fixed and may not be changed arbitrarily (Olivier, 1980:37). The following is an example of a typical entry, illustrating the order of the elements mentioned above:

UCT Libraries
Manuscript
BC312
WGA Mears
1826
1862
Donor: WGA Mears. These letters were copied and summarised by Mr. Mears from the originals among the Rev. Dr. John Philip’s papers in the Gubbins Collection, University of the Witwatersrand Library, which were subsequently in 1931 burnt in a fire in that Library; 18 items; TSS copies.

+ TSS summaries of letters by Sir Harry Smith and Sir Andries Stokenstrom to Capt. Caesar Andrews, 1835-1838, from the original MSS in the possession of N Sharpley.

+ TSS copies of land concessions in the O.F.S. by Moshesh and Sikonyela to the Wesleyan Methodist Church, 1833-1834, with a sketch map by WGA Mears.

+ Handwritten and TSS summaries of the following letters and correspondence as well as some TSS copies of entire letters: (i) Dr. John Philip - letters to and from John Fairbairn (1826-1862), 8, 26 & 18 pp. (ii) Freedom of the press (1825-1829) correspondence of Geo Greig, John Fairbairn, etc., 15pp.

**Fig 5.1 NAREM Entry**

(Gas, 1985:38)

NAREM is not based on a set of rules or codes. Hence the minimum description required is not clearly stated. In terms of the NAREM guide the title is optional and can be

248
ignored if it does not serve a purpose for a particular institution. The dates paragraphs are to be left blank if no period can be ascertained, nor is the summary mandatory (NAREM, 1985:4). In the light of the above situation, all elements are defined as MA except for the depot, reference number and type which are mandatory. The MA elements would depend on the participating institution, its policy and extent of detail required by it. Usually a collection has a title, stipulated period, a statement on the extent of the collection (remarks) and optionally a summary. Invariably, an entry has all elements except for the summary. The policy of GAS in accepting different levels of detail must be seen as an attempt to be flexible in accepting entries prepared by various kinds of participating institutions.

5.7.1 Depot

The Depot is the 'name of the institution where the collection is housed' (NAREM, 1985:4). The name consists of a short code to represent the institution, for example, 'Unisa' to represent the Unisa Documentation Centre. Unlike the Archives Depot where there is always a standard three letter code, in the case of other institutions, an intelligible form of their name, which is acceptable to the institution is given. The National
Register of Manuscripts, embraces all non-official documentation kept in both State Archives Depots and other institutions. This is culled together in a separate database called MAN (an abbreviation for 'manuscript'). The depot field is required to identify an entry in a network environment.

5.7.2 Type
Each institution may submit more than one type or category of material (maps, microfilms, manuscripts, etc.). Type 'manuscript' is mandatory for manuscript collections of participating institutions (NAREM, 1985:4). For private collections in an archive, type 'accession' apply (Olivier, 1980:37). Both the Depot and Type of material constitute part of a typical entry, enabling identification of an entry by it's type as well as the participating institution.

5.7.3 Reference
The accession number is the guiding principle for entry presentation as far as NAREM is concerned. This number is the reference and call-up number (Olivier, 1980:37). The Reference Number, while in addition to being used to locate items on the shelf, is also the chief identifier. Usually for a given type within an institution, separate
sequences exists, for example, for each type (manuscripts and microfilms) the following sequences - A1, A2, A3 ... and M1, M2, M3 ... may be used to represent different sequences\textsuperscript{6}. The Reference is sufficient to identify an entry uniquely within an organisation and forms the key to the collection.

5.7.4 Title

When a collection is received, the first step is to conduct a preliminary study in order to establish the main subject which is usually a person, subject or organisation. If the main subject is a specific person, irrespective of whether he is the donor or not, the person will be the subject of the preliminary study and his name will constitute the title of the collection. Where a person has collected documents about a wide variety of subjects, then the donor becomes the subject and the title of the collection (Olivier, 1980:39).

5.7.5 Dates

The dates include both the starting and ending dates, that is, the inclusive period covered by the collection. If the day and month are provided the date is written as follows 19381216 for 16 December 1938. Should only one
date be stipulated, it is entered in both date paragraphs. If no period can be deduced, these paragraphs are left blank (NAREM, 1985:4).

6.7.6 Remarks

The remarks paragraph contains optionally:

a. Name of donor;

b. Extent of item, that is, volume of donation, shelving space, number of volumes, linear metres etc.;

c. Placement property, for example, when placed with another donation (Olivier, 1980:38);

d. Existence of any finding aids, for example, inventories, and lists;

e. Access restriction on usage or conditions of donation; (NAREM, 1985:4) or any copyright restrictions that may apply (Smith, 1978:28).

The remarks paragraph, even though it specifies the above elements, seems to be an open book regarding what should be included. For the sake of completeness a 'miscellaneous' category is introduced to serve a utility function in accepting undefined and unanticipated elements in this paragraph (see Fig 5.2).
5.7.7 Summary

According to Olivier, the summary reflects the arrangement and content of a collection based on chosen groupings or headings (Olivier, 1980:39). The summary has two components. The first component consists of a brief explanatory note about people, places or events relevant to the accession. This is followed by a synoptic list of the content (files and documents) of the donation, in order of series (NAREM, 1985:4). The list of contents starts on a new line and is preceded by a ‘+’.

The entire set of elements in a typical NAREM entry consists of only one primary entity and a designate entity for the remarks. This entity is modelled as in Fig 5.3.
5.8 NAREM: Criticism and Evaluation

When compared to the AACR2R it is noted that each NAREM paragraph contains zero, one or more elements of the AACR2R. According to Marie Olivier, Chief Archivist for the Government Archives Services, the punctuation of the new library cataloguing code, referring to the AACR2, will not affect the paragraph style and will therefore not cause any problems. ‘The idea is not to standardize in such detail that institutions will have to abandon tried and customary methods and vocabulary’ (Olivier, 1980:37).
Control as far as NAREM is concerned, is at the repository and collection levels. The guide forms the chief mechanism of control. At the lower levels, control is left to the institutions concerned. These may consist of inventories, lists, and catalogue cards. A system is judged by its ability to accommodate both simple and complex record formats. Such records must extend beyond the collection level. By having only collection level control means that at the national level, key manuscripts lower than the collection level are not brought under control. Further, there is no provision for recording the hierarchical structure of a collection. However, this may be documented in the summary field of each entry, that is, hierarchical links are described in the summary rather than recognised in the database itself (Merrett, 1990:4). In a hierarchically linked database independent records are created for each unit catalogued which are in turn linked to a higher and/or lower levels (that is, parent to child and child to parent).

The system is not geared towards controlling individual files and items, nor is there a facility to produce inventories for individual collections or parts of it. The lack of sophisticated means for retrieval of important documents within the collection means that a
local system is essential to retrieve such documents and to produce inventories, indexes and other finding aids which are the responsibility of the local institution. The contributing libraries are expected to have a complementary means of conducting in-depth retrieval and finding the documents in a collection (Merrett, 1990:4).

NAREM descriptions are essentially free format allowing more than one element per paragraph. In the remarks paragraph, for instance, details on the donor, finding aids, previous donation etc. are recorded. The unstructured approach to entering and recording data makes element identification by machine-readable means difficult. It is difficult to extract a particular element, for example, to isolate a finding aid or condition of donation for instance, in an unambiguous way to facilitate exchange. No rules are prescribed, hence the order of the elements within the paragraphs are not controlled making it difficult to use automated means to extract desired elements within the paragraphs.

One of the major drawbacks of NAREM, is a lack of standardisation for each element. No authority file or thesaurus is available to control the usage of terms. Even though a library may establish a policy to

256
standardise the terms used, it is unlikely to tally with that of any other institution (Merrett, 1990:5). This means that standardisation is only helpful as far as the participating institution is concerned and not over the entire database, which comprises of different standards and conventions. A lack of standardisation affects retrieval.

Retrieval is achieved through the use of STAIRS (Storage and Information Retrieval System) software. Every significant word becomes a keyword with up to 16 verbs to manipulate and search the databases. Search on keywords, character strings, one or more paragraphs, use of linkages, synonyms and methods to narrow the search are considerable. While the method of retrieval is sophisticated, a free text language approach means that considerable intellectual effort is expended by the user in framing search strategies (Merrett, 1990:5) and weeding out unwanted entries.

According to Merrett (1990:10), NAREM is old fashioned, dependent upon a dataline, crude in the area of retrieval with high recall and low precision. The local institution is dependent upon policy decisions taken by an external agency. The lack of in-depth cataloguing, and authority
file control are added disadvantages. However, despite these criticisms, the various guides produced for the archive and non-archive services by the Government Archives Services, plays an important role in national bibliographic control of archives and manuscripts in the country. Terminals are available at the major centres such as Pietermaritzburg and Pretoria to tap online into the various databases of the various archival depots, documentation centres and special libraries within the country.

Today some national agencies use library techniques to describe government publications as discrete items, while it is not common for large libraries and documentation centres to use archival techniques. These point to the fact that library or archival techniques are not the province of libraries or archives respectively.

While acknowledging that two distinct traditions and practices exist, a unity of descriptions is possible. In the next chapter a composite model is developed by combining the descriptive elements outlined by the AACR2R and NAREM. The fundamental reality is that libraries as well as archives keep manuscripts and as such a unified model would serve both kinds of
institutions if it can be demonstrated that data elements in an integrated database can satisfy the needs of the two professions.

**NOTE**

1. Both manuscripts and manuscript collections have been discussed in the previous chapter of this thesis.
2. Changes in the organisation, chronology, subject, functions or filing schemes, influence the determination of a series.
3. Discussed under section 5.3.4 below.
4. See section 5.7 for an example of a typical NAREM entry taken from a NAREM guide.
5. Note two levels are defined in the note, that is, one the collection and the other, the papers of which it is a part.
6. However, a continuous numbering system can also be used in integrated systems and the type itself identified by a separate code, see GMD codes (Appendix A) defined in AACR2R.
CHAPTER 6
COMPOSITE MODEL

6.1 Introduction
The study was based on the premise that descriptive elements of manuscripts, whether in libraries or archives, are fundamentally the same; that it is possible to develop a composite model to integrate the elements found in typical AACR2R and NAREM entries. It was hoped that in taking such an approach, the study would support a unitary approach to the cataloguing of manuscripts alongside published materials, both of which have similar descriptive characteristics.

'The effort to organize, describe and make accessible a body of materials is a common cause of librarians, archivists, curators and other information professionals. The approaches used by each group vary with the nature of the materials and the traditions and purposes of those who manage them' (Fox, 1990:17). Although bibliographic and archival records are each derived in a different manner, both serve the same surrogate role, conveying to the reader an understanding of the nature of the
original. Fortunately, there are common elements that permit the integration of these different types of descriptions into one catalogue.

While varying in detail, catalogues and other finding aids are constructed according to basic descriptive principles, which use data inherent in the material itself (Smiraglia, 1990:5). While in libraries, the catalogue is the primary finding aid, in archives it is just one of the many tools employed, amongst others (indexes, lists, inventories etc.). In their descriptions, common data, such as titles, primary responsibility (main entry or provenance), extent of item and dates are the same, even though the content, rules and emphasis may differ. In such an integrated catalogue, all related materials may be obtained through a single search, even though the techniques of searching, the aids used and the types of materials encountered may vary in both practices (Fox, 1990:18-19). Both goals are to exploit the recorded knowledge. The objective of bibliographic and archival description are the same; only their priority and ranking differ.
Historically, librarians and archivists have emphasised the differences in the nature of the materials housed and techniques used. With the advent of the Machine Readable Cataloguing: Archives and Manuscripts Control (MARC:AMC), archivists have shown considerable interest in library cataloguing, indexing practices and related standards. From 1984 to 1986, twelve libraries participated in the AMC RECON project (funded by the National Endowment for the Humanities (NEH) Research Resources Program and the Pew Memorial Trust), converting approximately 21,000 archival and manuscript records, into the Research Libraries Information Network (RLIN) database, essentially a bibliographic database. Knowledge of the AACR2 and the APPM were mandated as standards for cataloguing (Cloud, 1988:574). Details supplied for each record included the main entry, title, date, physical description and a code to identify the participating institution. In the light of above developments 'descriptions of bibliographic and archival material have become compatible and interactive' (Madden, 1991:48).

In a study conducted by Hesselager (1984:260), he compared the kinds of cataloguing practices and techniques used in archives and large libraries and concluded that there was little difference in the end.
Some archives described government publications individually according to library principles. In the area of techniques too, while libraries dealt mainly with publications and archives with records, the overlap was not clear, for example, manuscript libraries using archival techniques and archives using library techniques.

In the light of the above arguments, a unitary approach using an integrated catalogue was mooted. However, before this could be achieved, the approaches taken in this study require clarification and elaboration.

6.2 Composite Model Development: Approach taken in Study
This study is biased towards the principles and practices as adopted by the library profession. Two factors mitigated this bias. Firstly, manuscripts are considered as just one form of library material. Secondly, the rigid rules as laid down by the library practice specify the inclusion or exclusion of specific elements in the description, in a definite order using prescribed punctuations. Hence the BM model developed for the AACR2R in the previous chapter of this thesis was used as the basis and point of departure for developing a composite model. NAREM elements were then added and reconciled with
the existing BM model while taking cognizance of deviations in the role and function of the various NAREM and AACR2R elements. However, before a composite model could be developed numerous decisions had to be taken outlining the approach taken in the study, the elements used in the descriptions, reporting, level of description to be used etc. The composite model arising from the combined NAREM and BM models were based on these decisions and considerations.

6.2.1 AACR2R And Modelling

The AACR2R was used as the basis for model development. All NAREM elements were then incorporated into the BM model of the AACR2R. However, not all the test data conformed to the AACR2R. Data was taken from each of the practices (AACR2R and NAREM entries) including hybrid practices such as the APPM to enable fuller testing. The Documentation Centre (University of Durban-Westville) entries were, however, compatible with both practices. Further, only elements present in the test data were included. For instance, no attempt was made to construct a heading if one was not provided. While the rules were followed in many of the examples, no attempt was made to ensure that the correct rules were employed in the test data. They were accepted as is. However, where deviations
in the use of the code was detected, for example use of 's.l. : s.n.' in the date area, they were omitted from the description. Emphasis was placed on the format and punctuation which served as indicators to the rules and elements used. The model thus served as containers to the data with due consideration for the rules used. It was hoped that the resulting model would be able to accommodate variations in practice of participating institutions.

6.2.2 Completeness of Description
All details were taken from the source documents. Neither the AACR2R nor NAREM were expected to be complete models for the various descriptive elements. The composite model was designed to accommodate all elements through the extensive use of the notes, for example, a note such as 'This is an experimental record', while meaningless, was included as a miscellaneous note. The notes served as a useful utility field.

6.2.3 Level of Description
Manuscript collections are catalogued at the highest level. The principle of levels correspond with the bibliographic concept of analysis (discussed in chapter 4 of this thesis) whereby descriptions are prepared for the
part of an item for which a comprehensive entry has been made (Hensen, 1989:5-6). In this study analytic entries were made for selected documents which linked the item(s) to the collection to which they belonged. The use of analytics for selected documents provided a reasoned balance between mass cataloguing of records on the one hand and the cataloguing of all items discreetly, on the other hand.

The 'in' analytic defined for all types of materials in the AACR2R is inadequate for manuscripts reflecting more than two levels. In terms of the rules, the 'in' analytic entry refers to the level immediately above it in the hierarchy. Hence no indication is given of the collection from which the item comes from when the level of the item is greater than the second level (the second level automatically refers to the first level description when describing the whole). Also there is no indication of the part of the collection being described, for example, items 26 to 30 in a collection of 100 items. Therefore for the sake of completeness of description and to point the user directly to the collection, a relationship complexity note was devised consisting of two parts, one the collection title and any numbering, classification or accessioning system that indicated the part being

266
described. Titles were, however, omitted if the part referred to the collection level title which was the same as the title in the 'in' analytic note.

Examples of Relationship complexity note:
Bhana Collection. 957/3
Mayat Collection. 4/15-31

In the first instance the item with number 957/3 is the only item that is being described and forms part of the Bhana Collection. In the second case items 15 to 31 are described as a group and are ordered at positions 15 to 31 within the Mayat collection. The cataloguer is responsible for maintaining the hierarchical relationships, which must be preserved by analytical entries and appropriate relationship complexity notes.

There may be several levels of description for manuscript material. What is important is that there should be a description at the most comprehensive level if there are to be additional records at any subordinate level. The number of levels in the study was, however, not prescribed. The choice of levels appropriate to individual collections was left to each repository or institution based on their own needs (Hensen, 1989:6).
While the levels were not prescribed, three basic levels, the collection (series), file and document/item levels were used for illustrative purposes. These are evident in the samples taken from the Documentation Centre and the Killie Campbell Africana Library.

The elements included in the description (analytics) with respect to manuscripts are:

**Elements of the Part**
- Title and statement of Responsibility
- Edition
- Date(s)
- Physical description
- Notes

**Elements of the Whole**
The above details were followed by the citation of the whole item which consisted of the:

1. Title and
2. Statement of Responsibility (optional)

The whole is described as an ‘in’ analytic note. In a typical catalogue entry, the ‘in’ analytic appears first and this is followed by a relationship complexity note to complete the description. Both elements appear in the note area of the part (AACR2R, 1988:300).
6.2.4 Identification

Each unit catalogued was assigned a unique bibliographic record number (BRN) that was automatically generated by the system. Through the use of the BRN a uniform method of identification was achieved even though each entry may have had a Reference number or local BRN which identified the item uniquely within a Depot or institution.

6.2.5 Reporting

Two basic types of reports as required by NAREM and the AACR2R were produced. Reporting is dealt with in detail in chapter 8 of this thesis. No provision for inventories, indexes and other finding aids were made. For NAREM purposes only collections were identified and isolated in the database (identified by a logical field called NAREM). When producing a library catalogue, all items at all levels, were included.

6.2.6 General Material Designation

Only manuscripts were dealt with in this study. Hence the GMD was omitted from the description.
6.2.7 Classification (Shelf Arrangement)

No specific kind of shelf arrangement was prescribed. Institutions may arrange items by accession and then by classification numbers using an internal classification scheme. Alternatively, a broad system of classification (for example, the Dewey Decimal Classification) may be used. Whatever the arrangement, the collection or parts of it should be located. Each of the BRN, Reference number and classification number were used for these purposes. Their varying functions and roles were preserved in this study. The responsibility is on the participating institution to adopt an acceptable method. For example, use of a number such as 89/2/12-35 can be construed as a means of classification, ordering or accessioning but in all instances the numbering point to documents 12 to 35 within the subsection '2' of the 89th collection.

6.3 Model Reconciliation: NAREM and AACR2R

The major inputs to the implementation design are the conceptual data model, the DBMS characteristics, and user processing requirements, for example, usage patterns, response times, backup and security parameters (McFadden & Hoffer, 1991:304). The Dbase DBMS characteristics have
already been discussed in chapter 2. Processing requirements have been discussed in chapters 4 and 5 and would be discussed further in chapter 7 when considering reporting. The two E-R model’s for the NAREM and the AACR2R entries developed thus far need to be combined to form the composite model. Thereafter, the resulting composite model requires translation into relational tables which are then subject to fine-tuning and modification. These are some of the aspects discussed in this chapter but first each of the NAREM elements are examined in turn and are merged into the existing AACR2R (BM) model.

6.3.1 Reference and Classification
The Reference number has two properties, one of identification and the other of ordering on the shelf. In library systems the property of ordering is accommodated by a classification number and identification is accomplished by a unique identifier (bibliographic record number) in computerised systems. Manuscript collections, as with NAREM participants, for instance, utilise the reference number to order the collection which is invariably the same as the accession number.

271
Libraries having both manuscripts and book-type material may not follow the latter approach and instead use the accession number to keep control of the items received by the library to identify each book or item uniquely. Collections are usually not subject to browsing. A well structured shelf arrangement can reveal the subject content of a collection and facilitate the location of items on a shelf. Some institutions may want to use a uniform system of arranging books and collections. To facilitate the location of items on the shelf in a classified arrangement, an optional classification number was defined in the study. The bibliographic model was extended to include the Reference and classification numbers. This is modelled as in Fig 6.1.

**Fig 6.1 Bibliographic Model (Part) including Reference.**

**Classification number and BRN**
6.3.2 Title
An institution using the AACR2R would generally construct a title for each entry. Descriptions prepared for NAREM only may not necessarily follow the AACR2R rules when constructing a title. The title is optional for NAREM participants but in this study, it was considered mandatory. A statement of responsibility is, however, not normally included for collections as it forms part of the title or the main entry and may be omitted. However, a statement of responsibility may exist for unpublished individual manuscripts and the AACR2R rules apply. The distinctions in AACR2R between the title proper, the other title information or parallel titles were retained when constructing titles for NAREM. No additional elements arising from the NAREM titles, were evident. Hence no changes were necessary to the existing title entity of the BM.

6.3.3 Depot
The Depot is only required when an institution participates in a wider environment, otherwise the code is not necessary. The sample included data from more than one depot and hence the depot formed a field in the extended BM (shown in Fig 6.2a).
Ideally incoming depot codes should be validated against a set of known codes during the input and update processes. To accommodate this, a new control entity was defined which consisted of a Depot field (Depcode) and a description (Depdesc), for example, a code ‘Unisa’ with a description of ‘Unisa Documentation Centre’. Where a Depot code was not available for an entry, the source, for example, ‘UDW’ with a description of ‘Documentation Centre, University of Durban-Westville’ was used. Thus the entity served to identify the Depot or any reference work or source from which the entry was taken. The entity is mapped as in Fig 6.2b.

Fig 6.2a BM showing Depot

Attribute

DEPOT

DEPCODE

DEPDESC

Fig 6.2b Depot Entity
6.3.4 Type
The Type for manuscripts is generally 'manuscript'. However, distinction is made between manuscripts by participating institutions and those held by the government archives, in which case the term 'accession' is used. These are the two possible values for the Type field. While in some instances the Type corresponded to the GMD, the Type was not used as a substitute for the latter. The meaning of the Type was restricted to the definition as specified by NAREM.

6.3.5 Dates
The NAREM dates correspond with the Date in the date area of the ISBD. Both the starting and ending dates are accommodated by the attributes Date1 and Date2. Hence, the Dates remained unchanged in the BM model.

6.3.6 Remarks, Summary and Notes
The Remarks paragraph consists of data from two areas of the ISBD. The extent and dimension in the Remarks paragraph correspond to the physical description area. Notes on any language, published version of a manuscript, donor and access restriction is already included in the notes area of the AACR2R. However, notes on any finding
aids or biographical information are new notes and required extensions to the basic BM model. A relationship complexity note and an ‘in’ analytic note were also included to accommodate levels of description. The structure of the notes as defined in the BM model was unaffected. Indicators were used to identify the type of note and adding new notes meant defining new indicator values (see Fig 6.3).

The Summary paragraph in a NAREM entry corresponds to the Summary in AACR2R (note area). A unique indicator, ‘Q’, identifies a summary. The presence of the ‘+’ character in a NAREM Summary is not part of an AACR2R summary and ought to be removed when constructing the summary for an AACR2R entry. This was accomplished by searching for the ‘+’ characters and removing them from the summary during reporting.

Considering that the sample was taken from different sources and that the notes were constructed as deemed fit by the participating institution, it was anticipated that not all the notes may be accommodated by the composite model defined thus far. Hence provisions were made for two miscellaneous notes, one for the AACR2R and the other for NAREM. The indicators and their descriptions are
shown below (Fig 6.3). The comments indicate AACR2R (for which the rules are cited) and NAREM usages as well as additional requirements (other). The notes as given below reflect the order of precedence. According to rule (4.7B) (AACR2R, 1988:133) a note is given first if it is decided that it is of primary importance.

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Type of Note</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>In Analytic</td>
<td>Chapter 13 (AACR2R)</td>
</tr>
<tr>
<td>B</td>
<td>Relationship complexity</td>
<td>OTHER 2</td>
</tr>
<tr>
<td>C</td>
<td>Nature, scope or form</td>
<td>4.7B1</td>
</tr>
<tr>
<td>D</td>
<td>Language</td>
<td>4.7B2</td>
</tr>
<tr>
<td>E</td>
<td>Source of Title.Proper</td>
<td>4.7B3</td>
</tr>
<tr>
<td>F</td>
<td>Variation in Title</td>
<td>4.7B4</td>
</tr>
<tr>
<td>G</td>
<td>Parallel &amp; other title information</td>
<td>4.7B5</td>
</tr>
<tr>
<td>H</td>
<td>Statements of Responsibility</td>
<td>4.7B6</td>
</tr>
<tr>
<td>I</td>
<td>Donor, source, previous owner(s)</td>
<td>4.7B7/NAREM</td>
</tr>
<tr>
<td>J</td>
<td>Place of Writing</td>
<td>4.7B8</td>
</tr>
<tr>
<td>K</td>
<td>Published versions</td>
<td>4.7B9</td>
</tr>
<tr>
<td>L</td>
<td>Physical Description</td>
<td>4.7B10/NAREM 3</td>
</tr>
<tr>
<td>M</td>
<td>Accompanying Material</td>
<td>4.7B11</td>
</tr>
<tr>
<td>N</td>
<td>Dissertations/Thesis</td>
<td>4.7B13</td>
</tr>
<tr>
<td>O</td>
<td>Access and Literary Rights</td>
<td>4.7B14/NAREM</td>
</tr>
<tr>
<td>P</td>
<td>Reference to published Descriptions</td>
<td>4.7B15</td>
</tr>
<tr>
<td>Q</td>
<td>Summary</td>
<td>4.7B17/NAREM</td>
</tr>
<tr>
<td>R</td>
<td>Contents</td>
<td>4.7B18</td>
</tr>
<tr>
<td>S</td>
<td>Ancient, medieval and Renaissance ms.</td>
<td>4.7B23</td>
</tr>
<tr>
<td>T</td>
<td>Finding Aids</td>
<td>NAREM</td>
</tr>
<tr>
<td>U</td>
<td>Biographical/Historical</td>
<td>OTHER</td>
</tr>
<tr>
<td>V</td>
<td>Placement with other Collections</td>
<td>NAREM</td>
</tr>
<tr>
<td>W</td>
<td>Miscellaneous</td>
<td>OTHER</td>
</tr>
<tr>
<td>X</td>
<td>Miscellaneous</td>
<td>OTHER (NAREM)</td>
</tr>
</tbody>
</table>

Fig 6.3 Notes included in the BM
Fig 6.4 E-R Diagram of Bibliographic Model (BM)
The new composite model (BM) is shown in Fig 6.4. All NAREM paragraphs are accommodated in the model. This includes the Reference, Depot, Type, Title, Dates, Summary and Remarks. In addition, the model includes elements such as the classification number, BRN and miscellaneous notes that did not form part of the NAREM or AACR2R models but were included for the sake of completeness.

6.4 Mapping Attributes onto Relational Tables

Once all E-R models have been combined into a composite model, the next step in the design process is to map the E-R model onto relational tables. The field names, types of data elements (numeric, logical, character), their lengths (fixed or variable) and the maximum number of characters allowed are then defined. Tentative structures for the relation tables were subsequently defined. Illustrated below (Fig 6.5), is the primary table (called Table1).

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRN</td>
<td>Bibliographic Record Number</td>
<td>N</td>
<td>10</td>
</tr>
<tr>
<td>Ref</td>
<td>Reference Number</td>
<td>C</td>
<td>204</td>
</tr>
<tr>
<td>Depot</td>
<td>Depot/Source</td>
<td>C</td>
<td>20</td>
</tr>
<tr>
<td>Type</td>
<td>Manuscript/Accession</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>Main</td>
<td>Main Entry Heading</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>Narem</td>
<td>NAREM Collection Indicator</td>
<td>L</td>
<td>1</td>
</tr>
<tr>
<td>Class</td>
<td>Classification Number</td>
<td>C</td>
<td>20</td>
</tr>
</tbody>
</table>

Fig 6.5 Table1
In the above table, the BRN is the key. The Ref, Depot and Type correspond to the elements as defined by NAREM. The Main field consists of the main entry heading while the Class field is reserved for a classification number. The Narem field indicates that the entry is to be reported to NAREM.

The following tables are designate/characteristic entities as identified in chapter 4 of this thesis. They are the title, responsibility statements, edition, date, physical description and notes. Each tuple in tables 6.6a to 6.6g is uniquely identified by the BRN and is the same as the BRN in Table 1 for the same entry. In this way the dependency between the primary table and the designate/characteristic entities were preserved. In some cases further dependencies were observed. For example, in the case of the responsibility statements, (Resp Table), in which case an additional key was defined. In the Resp table both the BRN and an additional key, (Tkey), for the Title to which it was linked, was included.

<table>
<thead>
<tr>
<th>BRN</th>
<th>Key for Title</th>
<th>N</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_ind</td>
<td>Title Type</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>Title</td>
<td>Title</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>Tkey</td>
<td>Title/Resp Key</td>
<td>N</td>
<td>9</td>
</tr>
</tbody>
</table>

_Fig 6.6a Title Table_
The Title table (Fig 6.6a) includes the title proper and one or more parallel titles. A separate field was not defined for the other title information. For each title, the title proper was separated from the other title by a ' : '. A unique BRN linked each title to the entry with the same BRN in Table1. The T_ind field indicated the type of title (title or parallel title). The Tkey field, on the other hand, formed a link between the Resp table (Fig 6.6b) and Title Tables linking each responsibility statement to a unique title within a particular record. The tables for the edition (Fig 6.6c) and its associated responsibility (Fig 6.6d) were defined, similarly. The date (Fig 6.6e), Physical description (Fig 6.6f) and notes (Fig 6.6g) were defined as well.

| BRN | Key for Resp | N | 10 |
| Resp | Responsibility Statement | C | V |
| Tkey | Title/Resp Key | N | 9 |

**Fig 6.6b Resp Table**

| BRN | Key for Edition | N | 10 |
| Ekey | Edition/Resp Key | N | 9 |
| Ed | Edition Statement | C | V |

**Fig 6.6c Edition Table**

| BRN | Key for Edition-Resp | N | 10 |
| Eresp | Edition/Resp Statement | C | V |
| Ekey | Edition/Resp Key | N | 9 |

**Fig 6.6d Resp2 Table**
All Depot codes in Table1 required validation to ensure that the correct Depot code was recorded in the database (already discussed in section 6.3.3 of this thesis). A table (Fig 6.6h) consisting of a depot code and description was therefore defined as part of the model.

### Fig 6.6h Depot Table

<table>
<thead>
<tr>
<th>Depcode</th>
<th>Depot Code</th>
<th>C</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depdesc</td>
<td>Description of Code</td>
<td>C</td>
<td>50</td>
</tr>
</tbody>
</table>

### 6.5 Denormalisation

Through the process of normalisation, large files are reduced to smaller manageable ones. According to Date (1986:461), the objective of normalisation is to reduce redundancy. Once dependencies and repeating groups are
removed, normalisation should stop. However, there is a danger of overnormalisation. Ullman (1982:234) warns that excessive decomposition may cause a loss of some dependent relationships.

The tables defined above are fully normalised. A fully normalised design is wasteful and expensive to implement as evident from the above tables. There is essentially one bibliographic entity with many dependent entities. The dependent entities share a common BRN and always occur together. Any updates, additions or deletions to the bibliographic entity or parts of it may be executed via a single key. To alter a title, publisher or note, for instance, the BRN or key must be provided to the system before access is gained to the respective elements for update or additions. Further, changes to these elements do not affect similar elements of any other record in the system and are therefore independent of them, that is, the integrity of the database is kept intact when performing any database operations on one particular entity. Anomalies are checked and controlled in such an environment. Anomalies would occur only when entities are shared across bibliographic records, which is not the case in this study. The elements are used only

283
as bibliographic descriptions, and not as access points, which would normally require authority files that relate elements across bibliographic records.

According to Date (1986:462-463), fully normalised relations may be denormalised to produce one entity for the purposes of retrieval and update. He asserts that if two items of information are very frequently retrieved together and are only very infrequently updated, as a design guideline, denormalisation is acceptable if there is a clear advantage for doing so.

The subordinate entities (Tables 6.6a to 6.6g) have been identified as designate entities of the ISBD. Date (1986:438) remarks that designate entities are not normally mapped into separate tables. He adds further that 'Full normalization followed by selective denormalization may lead you to consider design possibilities that would never have arisen if you had simply stopped short during the original normalization process' (Date, 1986:463). Based on the above considerations, four major reasons are advanced for the denormalisation of tables 6.6a to 6.6g. These are:

- the dependent entities are designate.
- the current design is wasteful.
- all elements of the bibliographic entity can be accessed as a unit.
- no anomalies would arise if the designate entities are grouped with the primary bibliographic entity (Table1) as the elements are not shared across bibliographic records.

Denormalisation of the files was accomplished by using the internal punctuation as prescribed by the ISBD. The titles and statement of responsibility were grouped together. This preserved the original order (dependencies) of the elements and the one to many relationships between the title and other title information, the parallel titles and other title information as well as the various statements of responsibility for the different types of titles. In addition, the cataloguing rules for works with no composite title were preserved as the order of the elements within the area were kept intact. Similarly, one field for the edition and its subsequent statements of responsibility using the prescribed punctuations, achieved the same results as the title. By using the
internal punctuations for the physical description all elements within the area were accommodated and accessible as a unit.

The dates however, were problematic. The date format includes variations such as '12 May 1965', '19--' and '196?'. These were not translatable into equivalent numeric fields or date types in Dbase. Hence a character field (Dates) was defined for both single and inclusive dates. The field is unformatted and therefore allows for wide variations in date formats.

NAREM requires a rigid format for the opening and closing dates. The Dbase format mm/dd/yr (for example, 12/30/92) is not adequate for NAREM dates which has the format dd/mm/year (for example, 30/12/1962). A separate character field of eight characters was therefore defined for each of the opening and closing dates (Date1 and Date2 respectively) to ensure that the data was in the required form. However, to force numeric data to be entered in the character fields, Date1 and Date2, a picture clause, namely, Picture '99999999' was used as a screen interface. The character, instead of a numeric field, was preferred because it is not uncommon to find alpha characters in such fields, for example, ca 1895
even though the rules stated otherwise. For the purpose of this study only exact dates were entered, for example, ca 1895 was entered as 1895. However, as a compromise, the date was entered as a free format date in the Dates field, if not already in the title. The fields, Date1 and Date2, required a range validation for the day (0 to 31) and month (0 to 12). The zero is an acceptable value since not all Dates have the day or the month supplied.

For every primary table entry, in terms of the new structures defined above, there is at most one field for the title, dates, edition and physical details. The notes on the other hand are repeatable and was therefore tentatively, kept in a separate table. Incorporating the entities as fields in Table1, yielded the following table:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRN</td>
<td>Bibliographic Record Number</td>
<td>N</td>
<td>10</td>
</tr>
<tr>
<td>Ref</td>
<td>Reference Number</td>
<td>C</td>
<td>20</td>
</tr>
<tr>
<td>Depot</td>
<td>Depot</td>
<td>C</td>
<td>20</td>
</tr>
<tr>
<td>Type</td>
<td>Manuscript/Accession</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>Main</td>
<td>Main Entry Heading</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>Title</td>
<td>Title and Responsibility</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>Ed</td>
<td>Edition &amp; Responsibility</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>Dates</td>
<td>Dates</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>Date1</td>
<td>Date (Opening)</td>
<td>C</td>
<td>8</td>
</tr>
<tr>
<td>Date2</td>
<td>Date (Closing)</td>
<td>C</td>
<td>8</td>
</tr>
<tr>
<td>Physdesc</td>
<td>Physical description</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>Narem</td>
<td>NAREM Collection Indicator</td>
<td>L</td>
<td>1</td>
</tr>
<tr>
<td>Class</td>
<td>Classification Number</td>
<td>C</td>
<td>20</td>
</tr>
</tbody>
</table>

*Fig 6.7 Table1 (Denormalised)*

287
The resulting table (Table1) is more compact. By using the ISBD punctuation, repeating groups are accommodated, dependencies are controlled and the order of the elements in the format is preserved. Thus far three tables constitute the database, Table1, the Depot table and Notel. Table1 and the Depot table are linked by a common Depot code.

Table1 (*BRN, Ref, +Depot, Type, Main, Title, Ed, Dates, Date1, Date2, Physdesc, Narem, Class)
Depot (*Depcode, Depdesc)
Notel (*BRN, Note_ind, Note_desc1)

6.6 Indexes and Indexing

When producing catalogues, indexes and lists, the entries are required to be in a pre-determined order. The collating sequence varies with the type of software used. The ASCII (American Standard Code for Information Interchange) collating sequence essentially sorts character by character and this does not conform to the word by word filing of the many filing codes such as the ALA filing rules, BLAISE filing rules and Library of Congress filing rules. Some programs are case sensitive and therefore do not ignore upper and lower case, for example, 'The story of O' files before 'The story of aviation' (Hunter, 1985:92-93). A special program may be written to implement the filing rules. In small files as
in the case of the sample data used in this study, the letter by letter approach used by Dbase suffices but this may not be acceptable for large catalogues (Tedd, 1979:75).

The inherent problems with filing and ordering goes beyond sorting by case, letter by letter or word by word. Programs are not able to detect a filing article and make value judgements unless specifically so designed. Programs may not always distinguish between 'A level physics' and 'A tale of two cities' and isolate in which instance a filing article occurs. The problem is further compounded across languages, for example, 'Die' in Afrikaans and 'Die' in English, the term is a filing article in the former case but not in the latter. To achieve a measure of control over filing articles, a separate field was defined for the filing article of the title. For a given title the complete title may then be reconstructed by concatenating the filing article with the remainder of the title. The title 'A tale of two cities' is reconstructed by using the following Dbase command Trim(A) & ' ' & 'tale of two cities' to yield the full title. Applying the same rule to 'A level physics'
yields 'A level physics'. Thereafter the Ltrim() function is applied to remove leading blanks as evident in the second example.

The above examples suggests that special rules for sorting and ordering may be required for the main entry headings. The titles, names, places, corporate bodies and uniform titles are used for validation and ordering of reports. It was therefore essential to define the rules for sorting and ordering the headings. The indexing requirements of each type of heading are examined and defined below.

6.6.1 Indexing on Titles
The standard Dbase indexing option that allows indexing on the entire title is not adequate to trace duplicate titles. While a longer title tends to identify the work unambiguously, providing a more distinctive title, it may have implications for machine searching and matching. The punctuation, long titles and case affect the degree to which an exact match can be found. Hence system designers have adopted various techniques for searching a title, for example, keyword searching, compression techniques or a combination of both (Tucker, 1990:243).
The technique of compression is particularly attractive in that an entire title can be reduced to a key of a fixed length. The key can thereafter be used to validate titles and test for a match. The Library of Congress uses the compression technique. The key is generated from the first three characters of the first word of the title proper followed by the first character of the next three words, yielding a total of six characters at the most. The initial article is ignored. The title 'The Don Juan theme, versions and criticisms' is reduced to the key 'donjtv'. The indexing as used by the Library of Congress, is sensitive to subtitles. When a ':' is inserted in the above title as follows 'The Don Juan theme : versions and criticisms', the key generated is 'donjt' (Tucker, 1990:243). In this study, the subtitles were ignored when generating keys as a result both the above titles would compute to the same key, 'donjtv'. To accommodate the key, a new field, Tkey, of length 6 characters, was defined for Table1. To facilitate access, an index, Tkeyndx, was defined on the field Tkey. To solve the problem of isolating and identifying the filing article, a separate field, T_art of 10 characters was defined. To achieve uniformity all titles in reports were ordered in upper case, the filing articles were excluded.
6.6.2 Indexing on Names

The failure to integrate heading structure with filing codes, for example, the placement of 'Sir' in a name heading prior to the forename, even though 'Sir' is ignored in filing, implies that there is some hidden sort key, some other text or key used for arrangement. The heading or main entry is not necessarily an arranging element. At best the main entry heading serves as a basis for the sort but is not necessarily the sort key (Carpenter, 1989:82-84).

Author headings were used to validate new names against existing names in the database and sort reports by names. The entry element (which is usually the surname) and the forenames were considered the most important elements in a name. To increase the possibility of a match, the initials instead of the forenames were used to form part of the key (Mkey). Fifteen characters were allocated for the key which included an '*' and at most two initials. The results of the key generation algorithm are illustrated in the examples below. The algorithms to generate the key is found in the procedure file, Appendix D. Procedure names calls procedure Stff which removes unwanted punctuations from the entry element. Procedure wcomp2, called by procedure names, strips the initials

292
into separate identifiable words and passes the initials Sarr1 and Sarr2 back, which are then combined with the entry element to yield the key. The key is constructed using the following rule: Mkey = Entry Element + '∗' + First Initial + Second Initial.

<table>
<thead>
<tr>
<th>Names</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Ram Gopal</td>
<td>RAM GOPAL</td>
</tr>
<tr>
<td>2. X, Malcolm</td>
<td>X*M</td>
</tr>
<tr>
<td>3. Day-Lewis, C</td>
<td>DAYLEWIS*C</td>
</tr>
<tr>
<td>4. Gregory, Augustus D, Baron</td>
<td>GREGORY*AD</td>
</tr>
<tr>
<td>5. Stanhope, Lady, Hester</td>
<td>STANHOPE*H</td>
</tr>
</tbody>
</table>

It is evident from example 5 above, that titles of nobility do not form part of the key. The algorithm aught to isolate titles of nobility as well as any other element that does not form part of the key. In order to achieve this each of, the entry element, forenames, titles of nobility and additions required a separate identity. Five fields, Entry, Fore, Nobility, Adds and N_other were defined. The additions field included dates, fuller forms of a name and distinguishing terms (AACR2R, 1988:408-417). N_other was used to accommodate elements such as translators, compilers and other relator codes.

Names are of variable length and some means is required to store the names efficiently. The main entry heading for names was stored using the rule below:
The above format is ideal to store variable length data. The delimiter, Asc(181), served to separate the various components. It is observed in the above examples that the titles of nobility may follow the entry element (example 4 above) or the forenames (example 5 above). The latter usually applies to British terms of honour (AACR2R, 1988:408-409). A new field (British) was defined to reflect this position and any other instance requiring the title of nobility to appear after the entry element. This information was recorded with the name heading, the format of which now becomes:

Another field was required to identify the type of main entry heading so that only relevant types of headings could be presented during validation. A field named Mtype was defined. Whenever the main entry heading was a name the field had a value of 'A'. This field was included as a separate field in table1 so that it could be indexed for fast retrieval.
When producing a report under main entry for names, the entire name required reconstruction. The following order of precedence: entry element, nobility (if applicable), forenames, nobility (if applicable), additions, and other information (N_other) was used. To achieve proper ordering and sorting, the entry element and forenames were assigned 50 characters while the Adds and Nobility were assigned 30 characters each. An index for the name was constructed on the entry element and forenames. The format is illustrated with examples, below. By assigning 50 characters to the entry element and forenames and padding the remainder of the field with blanks, the names were indexed correctly. See section 6.6.4 for index construction of name headings that are combined with uniform titles.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Forenames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>A Thomas</td>
</tr>
<tr>
<td>Smithy</td>
<td>T T</td>
</tr>
<tr>
<td>Smythe</td>
<td>A T</td>
</tr>
</tbody>
</table>

6.6.3 Indexing on Place Names and Corporate Bodies

In the study Geographic names were divided into two segments, the place and additions (examples 1-3 below). Both the place and additions were included in index construction. For corporate bodies, a main body, a
subordinate body and any additions included in the heading (examples 4-7) were defined. While in the former case the additions included qualification of the place, in the case of corporate bodies, the additions included the date, place and conference data. The design supported more than one subordinate heading. This was achieved by separating each subordinate heading with a full stop and space. The heading and subordinate headings were assigned a total of 100 characters while the additions were assigned 30 characters and this formed the basis of the index entry for the place and corporate bodies.

For the purposes of validation, the keys were generated using the same rule for title key generation. Additions were excluded. Example of keys generated are shown alongside the headings below. Again, the Mtype field was used to identify the type of heading. The value of Mtype was set at 'C' for corporate bodies and 'P' for places. For the purposes of storage both places and corporate bodies were assigned the same variables. The rule 'Heading = Body & Asc(181) & Adds' defined the storage format.
Examples

1. New York (State)                Key
   NEWY
2. Saint Anthony (Hennepin County, Minn.) SAIA
3. Saint Anthony (Stearns County, Minn.) SAIA
4. Bounty (Ship)                   BOU
5. World Peace Congress (1st: 1949: Paris, France, Prague and Czechoslovakia) WORPC
7. United Kingdom. Prime Minister UNIKPM

Procedures for Names, Place and Corporate Bodies

In the discussion above, the variables and procedures have been over-simplified and meaningful names to the variables were given, for example, fore for forenames.

However, a uniform method was sought to store and extract the elements of the main entry heading using only one procedure. Procedure Conmain (Fig 6.8) was subsequently defined. It contains six elements, El1, El2, El3, El4, El5 and El6 which accommodates the 6 variable fields in a the heading. The fields are concatenated and passed as parameters to form the storage format. For each of the names, place and corporate bodies, the position of the elements within the algorithm differ. Their positions are set within the procedures Names, Place and Corp (all in the procedure file, Procf1). The following command concatenates the fields for the Names is:

   Do Conmain with Element1x, Element2x, Addsx, Nobilx, Element3x, Britishx
where the elements are in the order, entry element, forenames, additions, nobility, relator codes, and British terms. Similarly, for the place and corporate bodies, only the entry element and additions were stored (E11 and E13) as shown below:

Do Conmain with Element1x, ",", Addsx, ",", ",", ".

Procedure Uconmain reverses the entire process by filling the variables E11, E12, E13, E14, E15 and E16 with the contents of the field Main. The procedure is very long and is omitted from the text but is found in Appendix D in the procedure file, Procfl.

Procedure Conmain
Parameters E11, E12, E13, E14, E15, E16
** Procedure strings 6 elements of the
** Main Entry Heading.
** Uconmain performs the reverse of this procedure.
Mainy = Chr(181) +Trim(E11) +Chr(181) +Trim(E12) +;
Chr(181) + Trim(E13) + Chr(181) + Trim(E14) +;
Chr(181) + Trim(E15) + Chr(181) + Trim(E16) + Chr(181)
RETURN

Fig 6.8 Procedure Conmain

6.6.4 Indexing on Uniform Titles
Two basic structures for uniform titles as defined by the AACR2R were defined. One consists of a heading only (example 4 below) and another consisting of a name,
place, title or corporate body, followed by a uniform title (examples 1 and 3 below). The names, places and corporate bodies in a uniform title, have been constructed in terms of the rules and algorithms defined above.

In terms of Rule 25.2 (AACR2R, 1988:484), uniform titles are enclosed in square brackets and appear before the title proper. Optionally, a uniform title is recorded without square brackets. In this study brackets were omitted.

**Examples**

1. Dickens, Charles Martin Chuzzlevit

2. Genesis (Anglo-Saxon Poem)

3. France Constitution (1946)

4. Bible. O.T. Genesis XII, I-XXV, II...
   (AACR2R, 1988:480-538)

**Key**

MARC

GEN

CON

BIBOTG

The key, ukey, for uniform titles utilises the same algorithm used for the titles. Additions were excluded in index construction. Two fields, Ubody of 100 characters and Uadds of 30 characters, were defined. A new field, Utype was defined to indicate the presence or absence of
a uniform title. The value of Utype was set at 'U', whenever a uniform title was present otherwise it was blank. For storage purposes the following rule applied:

Uniform = Uboby & Asc(181) & Uadds

As evident from the above examples, Uniform titles involve at least two levels of indexing. Where the first level of the heading included a name, place or corporate body, the rules for sorting for each heading as defined previously, apply. Where the heading had a second level uniform title, the field Uboby was sorted at a sublevel of such headings, thereby providing a two level sort as indicated below. Note in particular that the uniform title in example 3 below has no subordinate (level 2) index, thereby enabling 'Bible ...' to sort at the same level as 'Dickens ...'.

Dickens, Charles
France (1946)
Bible. O.T. Genesis XII, I-XXV, II...

Martin Chuzzlevit
Constitution

Uniform Indexing for Catalogue Production

Ideally, for sorting purposes, one index should be constructed over all headings. A separate field, Msort, was defined over which an index, Maindex, was
constructed. The index ordered the titles within the uniform titles which in turn was sorted within any given name, place or corporate body. Hence three basic levels were distinguished, level 1 which consisted of a place, name, corporate body, uniform title or title; level 2 which consisted optionally of a uniform title; level 3 the title. In this way all titles were sorted within the headings. Fig 6.9 illustrates the three levels which comprise of 5 columns, each having a standard width. The first element (entry element) was assigned 100 characters while the second entry element at level 2, was assigned 50 characters. Both the additions were assigned 30 characters each while the title was assigned 40 characters. All fields totalled 250 characters. Dbase is unable to accommodate 250 characters per index entry and therefore use was made of clipper indexes.

<table>
<thead>
<tr>
<th>LEVEL 1</th>
<th>LEVEL 2</th>
<th>LEVEL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTRY-ELEMENT ADDS</td>
<td>ENTRY-ELEMENT2 ADDS</td>
<td>TITLE</td>
</tr>
<tr>
<td>COLUMN</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>WIDTH</td>
<td>100</td>
<td>30</td>
</tr>
</tbody>
</table>

Fig 6.9 Main Entry Heading and Title Sort

301
Where no main entry was defined the first four columns were blank but where the title served as the main entry it appeared in columns 1 and 5. Likewise when no second level entry was present columns 3 and 4 were blank. By reducing a three level hierarchy to a flat index structure as defined above, uniformity when sorting, was achieved. In addition, by reducing all entries to upper case resolved discrepancies due to upper and lower cases. The new index fields were added to Table1 to yield the following table:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRN</td>
<td>Bibliographic Record Number</td>
<td>N</td>
<td>10</td>
</tr>
<tr>
<td>Ref</td>
<td>Reference Number</td>
<td>C</td>
<td>20</td>
</tr>
<tr>
<td>Depot</td>
<td>Depot</td>
<td>C</td>
<td>10</td>
</tr>
<tr>
<td>Type</td>
<td>Manuscript/Accession</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>T_art</td>
<td>Title Indicator</td>
<td>C</td>
<td>10</td>
</tr>
<tr>
<td>Mkey</td>
<td>Main Entry Key</td>
<td>C</td>
<td>15</td>
</tr>
<tr>
<td>Mtype</td>
<td>Main Entry Type</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>Main</td>
<td>Main Entry Heading</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>Tkey</td>
<td>Title Key</td>
<td>C</td>
<td>6</td>
</tr>
<tr>
<td>Title</td>
<td>Title and Responsibility</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>Utype</td>
<td>Uniform Entry Existence</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>Ukey</td>
<td>Uniform Key</td>
<td>C</td>
<td>6</td>
</tr>
<tr>
<td>Uniform</td>
<td>Uniform Heading</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>Ed</td>
<td>Edition &amp; Responsibility</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>Dates</td>
<td>Dates</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>Date1</td>
<td>Date (Opening)</td>
<td>C</td>
<td>8</td>
</tr>
<tr>
<td>Date2</td>
<td>Date (Closing)</td>
<td>C</td>
<td>8</td>
</tr>
<tr>
<td>Physdesc</td>
<td>Physical description</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>Narem</td>
<td>NAREM Collection Indicator</td>
<td>L</td>
<td>1</td>
</tr>
<tr>
<td>Class</td>
<td>Classification Number</td>
<td>C</td>
<td>20</td>
</tr>
<tr>
<td>Msordt</td>
<td>Main Entry Heading Sort Field</td>
<td>C</td>
<td>250</td>
</tr>
</tbody>
</table>

Fig 6.10 Table1 (After Including Index Fields)
In order to sort and retrieve data, numerous indexes were defined for both Table1 and the Depot Table. For the Depot Table, the index (Depot.ndx) was defined on the Depcode to validate incoming codes during the input and update processes. Entries were not limited to upper case only since some codes were in both upper and lower case.

For Table1 numerous indexes were defined. The index on the BRN, Table1.ndx, is the primary index providing access to the table by BRN. The index is used during any operation requiring random access to any one given record or groups of them. The Maindex index, as mentioned previously, sorted the main entry headings. The title key, using index Tkey, was used to validate the titles. Similarly, the main entry headings were validated by their type and key (Mtype + Mkey) using index, Mtkey. Uniform titles were defined similarly using the expression (Utype + Ukey) and index, Ukey. Finally, index Ref was defined on the Reference number which ordered reference numbers by their numerical value and optionally, a prefix. The key expression that defined the index is shown below:

Upper(substr(Ref,1,10)) + Str(val(substr(Ref,11,10)))).
6.7 Variable Length Fields and Clipper

Dbase caters for both fixed and variable length data. The files with dbf extension are used for the former while files with dbt extension are used for the latter. However, memo fields are not searchable or manipulatable. The fundamental drawback of using fixed length fields for variable data is that data may be truncated when the field length is too small or it leads to excessive wastage of space when the entire field is not used. A more improved design was sought that could allow for variable data to be accommodated and manipulated.

With Clipper, memo fields are searchable, and variables of up to 32k can be accommodated. For x and y where x is the variable and y the field name in the database, the command ‘replace y with x’ is used to update a memo field. When it is desired that the entire field be edited, the command ‘x= memo-edit (field_name)’ (Tiley, 1988:538), sets a variable x to the value of the field_name and allows the data to be modified using a pop-up editor. When data in the editor is changed the variable x is simultaneously updated. The editor is particularly ideal when dealing with long notes as in the case of NAREM abstracts which may exceed one page.
Dbase III+ cannot handle files more than four or five kilobytes. Anything longer cannot be saved and will be lost to the memo file meant to receive it (Beiser, 1987:71).

Dbase commands are compatible with Clipper program statements. However before the Clipper capabilities could be used by Dbase programs, the Dbase programs need to be compiled and linked. Compiling translates the Dbase code into machine code which consists of a series of ones and zeros. Linking enables the different programs and routines to act as one unified program. The two step process of compiling and linking are executed by Clipper commands run under DOS. For the file Aprog, where Aprog is the highest level program the command 'Clipper Aprog' compiles the program and all programs, procedures and subroutines below it. Linking is achieved by executing the command 'Plink86 FI Aprog, Ndx LIB Extend, Clipper'. The command links the files (FI), Aprog and Ndx and the libraries (LIB), Extend and Clipper to the application. When Dbase index files are used in an application, the Ndx files needs to be linked. The Extend and Clipper libraries contain all the utilities, routines and procedures defined for Clipper and as the routines are called in the application, they are included in the final
executable file (Aprog.exe). Aprog.exe can then be run independently of any host program under DOS. The following is a DOS batch file to achieve both linking and compiling.

```
CLIPPER APROG
PLINK86 FI APROG, NDX LIB EXTEND, CLIPPER
```

### 6.8 Fine-tuning using Memo Fields

The variable length fields in Table1 would usually be assigned one memo field each. This is wasteful as each field has an overhead of 10 bytes for each pointer per memo field and a minimum of one block (254 characters) is assigned at any one given time. Separately defined variable length fields for the fields in Table1, for instance, would result in considerable wastage of space if all 254 characters are not used in a block. At the level of physical design, emphasis is placed on fine-tuning the database. A solution to reduce wastage of space was achieved by culling together all variable length fields per record into one memo field. This reduced the number of blocks and pointers used per record.
In the field of data processing, a record format allows the discrimination between data elements when these elements have been translated into machine-readable form. The human is able to identify elements in a catalogue by their content and their position in the record format. A program on the other hand, must by some device have explicit identification of the data elements in a record to enable it to identify elements such as the author, title, edition and imprint of bibliographic records. Where this is not possible the format must introduce techniques which allow explicit identification of each data element (Knapp, 1968:276).

Two possible approaches, the directory and delimiter formats (discussed in chapter 3 of this thesis) can be used for variable length records. The study adopted the delimiter format. When the number of fields are few and the processing is limited (approximately 1000-1500 bytes per record), the delimiter format may be used. The delimiter format utilises special characters that do not form part of the data to serve as delimiters. The Asc(180) and Asc(181) characters were chosen to serve as delimiters.
The Dbase dictionary is able to only identify and locate the position of a memo field within the record. It is unable to specify the existence and order of the elements within the field when the field holds more than one element. Hence a format was devised to specify the existence and order of the elements within a memo field. By reserving a position for each element in the field (even if the values are null), computing and identifying individual elements is possible. A sub-dictionary definition of the variable length field (Mem) in Table1 (Fig 6.11) consisting of variable length fields, Main, Uniform, Title, Ed, Physdesc and Dates is defined below. This structure utilises the delimiter, Asc(180), in the format.

<table>
<thead>
<tr>
<th>Position</th>
<th>Segment</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Main Entry Heading Asc(180)</td>
<td>Main</td>
</tr>
<tr>
<td>2</td>
<td>Uniform Heading Asc(180)</td>
<td>Uniform</td>
</tr>
<tr>
<td>3</td>
<td>Title and responsibility Asc(180)</td>
<td>Title</td>
</tr>
<tr>
<td>4</td>
<td>Edition Asc(180)</td>
<td>Ed</td>
</tr>
<tr>
<td>5</td>
<td>Physical description Asc(180)</td>
<td>Physdesc</td>
</tr>
<tr>
<td>6</td>
<td>Dates Asc(180)</td>
<td>Dates</td>
</tr>
</tbody>
</table>

*Fig 6.11 Sub-dictionary Definition for Mem Field, Table1*
Notes

The above ‘Dictionary’ definition takes care of non-repeatable fields only. Repeatable elements within a field, for example, the notes have thus far not been catered for adequately in the design. In order to accommodate repeating fields, the notion of a group was introduced. Each group may consist of zero, one or more elements. By using the delimiter, Asc(181), at a sub-level of the above structure, the Note_gr can be represented as follows:

Asc(180)
Asc(181)
Note1
Asc(181)
Note2
........

NoteN
Asc(181)
Asc(180)

The group itself is delimited by a pair of Asc(180) characters within the general structure and each note is delimited by an Asc(181) character. The final structure for the variable fields within the memo field Mem, is illustrated in Fig 6.12. Table1 (Fig 6.13) was redefined to reflect the new structure.
### Fig 6.12 Memo Field (MEM) Structure within Table 1

<table>
<thead>
<tr>
<th>Position</th>
<th>Level</th>
<th>Subfield</th>
<th>Elements/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Asc(180)</td>
<td>Main Entry Heading</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Asc(180)</td>
<td>Uniform Title</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Asc(180)</td>
<td>Title &amp; Responsibility</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>Ed</td>
<td>Edition</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>Asc(180)</td>
<td>Physical description</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>Dates</td>
<td>Dates</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Asc(180)</td>
<td>(Note_gr begins after this delimiter)</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>Asc(181)</td>
<td>First Note</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Note2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Noteen</td>
<td>Last Note</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Asc(180)</td>
<td></td>
</tr>
</tbody>
</table>

### Fig 6.13 Table 1 (Final)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRN</td>
<td>Bibliographic Record Number</td>
<td>N</td>
<td>10</td>
</tr>
<tr>
<td>Ref</td>
<td>Reference Number</td>
<td>C</td>
<td>20</td>
</tr>
<tr>
<td>Depot</td>
<td>Depot</td>
<td>C</td>
<td>20</td>
</tr>
<tr>
<td>Type</td>
<td>Manuscript/Accession</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>T_art</td>
<td>Title Indicator</td>
<td>C</td>
<td>10</td>
</tr>
<tr>
<td>Mkey</td>
<td>Main Entry Key</td>
<td>C</td>
<td>15</td>
</tr>
<tr>
<td>Mtype</td>
<td>Main Entry Type</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>Tkey</td>
<td>Title Key</td>
<td>C</td>
<td>6</td>
</tr>
<tr>
<td>Utype</td>
<td>Uniform Entry Existence</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>Ukey</td>
<td>Uniform Heading Key</td>
<td>C</td>
<td>6</td>
</tr>
<tr>
<td>Date1</td>
<td>Date (Opening)</td>
<td>C</td>
<td>8</td>
</tr>
<tr>
<td>Date2</td>
<td>Date (Closing)</td>
<td>C</td>
<td>8</td>
</tr>
<tr>
<td>Narem</td>
<td>NAREM Collection Indicator</td>
<td>L</td>
<td>1</td>
</tr>
<tr>
<td>Class</td>
<td>Classification Number</td>
<td>C</td>
<td>20</td>
</tr>
<tr>
<td>Msort</td>
<td>Main Entry Heading Sort field</td>
<td>C</td>
<td>250</td>
</tr>
<tr>
<td>Mem</td>
<td>Memo Field (variable data)</td>
<td>M</td>
<td>V</td>
</tr>
</tbody>
</table>
6.9 Algorithms in Clipper

A variable length format guarantees compact storage. As evident in the Mem field, each element was assigned a unique variable and position. However, this was achieved at the expense of more complex programming. Two sets of algorithms were developed to compose and decompose the Mem field into their components.

The algorithm for concatenating the various component fields of the variable field Mem were accomplished by two subroutines, Connote and Conmem. The Connote routine accepts up to 24 notes and these are concatenated using the Asc(181) character as delimiter. Data for each field is gathered using variables that interfaces with the screen. The indicators are concatenated and stored with its associated note. The algorithm Connote is defined as illustrated in Fig 6.14.

Procedure Connote

** Concatenates the notes into one string using delimiter Asc(181)

Connotex = Space(1)
P2 =65
Do While P2 > 64 .and. P2 <89
   Ax = Ltrim(Str(P2))
   If Len(Trim(Sx&Ax)) >1
      Sx&Ax = Chr(&Ax) +Trim(Sx&Ax)  && Stores the Indicator
      Connotex = Connotex + Chr(181) +Sx&Ax
   Endif

311
P2 = P2 + 1
Enddo
Connotex = Ltrim(Connotex) + Chr(181) && Extra delimiter
        && added at end
RETURN

Fig 6.14 Subroutine Connote

After the notes are strung together, they are incorporated into the final variable length string, Conmem. Each subfield forming part of Mem field (see Fig 6.11) including the notes, are passed as parameters to the subroutine Conmem. Each field is separated by an Asc(180) character. The entire Mem field of Table1 is at this point fully constructed and ready for storage. The subroutine Conmem is illustrated in Fig 6.15.

Procedure Conmem
** Concatenates Mem field (level 1) for 7 subfields,
** using delimiter Chr(180)
** Fields are in order Main, Uniform, Title, Edition,
** Physical description, Date and Notes
Conmemx = Chr(180) + Trim(Mainx) + Chr(180) +
        Trim(Uniformx) + Chr(180) + Trim(Titlex) + Chr(180);
        + Trim(Edx) + Chr(180) + Trim(Phdescx) + Chr(180) +;
        Trim(datx) + Chr(180) + Trim(Connotex) + Chr(180)
RETURN

Fig 6.15 Subroutine Conmem

The subroutine Uconmem does the reverse of Conmem. Instead of concatenating the fields, each field is filled with the contents from the MEM field for each of Main, Uniform, Title, Notes etc. The Uconnote routine is then invoked to fill the contents of the 24 notes with data
from the note group (represented by the variable Connotex). Once the data is passed to the respective variables, they can then be modified and searched, if desired. Both the algorithms are shown as in Figs 6.16 and 6.17.

Procedure Uconmem
** Procedure Unstrings mem field using variable conmemx
** The Main, Uniform, Title, Edition, Physical
** Description, Date and Notes are filled into variables
** E1-E7.

If Clipper
Conmemx = Mem
Else
Conmemx = Space(1)
Endif
Loopcnt = 8
Private D_8
Private P_1
Private P_2
Private P_2X
Private H_1
Lword = 'F'
P_1 = 1
P_2 = 1
H_1 = "_"
E1 = Space(1)
E2 = Space(1)
E3 = Space(1)
E4 = Space(1)
E5 = Space(1)
E6 = Space(1)
E7 = Space(1)

**** Now Filling Each Field Using Chr(180) as Delimiter
If Len(Conmemx) > 1 .and. Substr(Conmemx,1,1) = chr(180)
   Conmemx = Substr(Conmemx,2,Len(Conmemx)-1)
Endif

Do While P_2 <= Loopcnt .And. Len(Trim(Conmemx)) > 0
   Conmemx = Ltrim(Rtrim(Conmemx))
If Chr(180) $Conmemx
   D_8 = Substr(Conmemx,1,At(Chr(180), Conmemx)-1)
   If Len(D_8) + 1 = Len(Conmemx)
Lword = 'T' && Determines if last word
Else && being processed
Conmemx = Substr(Conmemx, At(Chr(180), Conmemx) + 1,
Len(Conmemx)- At(Chr(180), Conmemx)-1)
Endif
Else
Lword = 'T'
D_8 = Substr(Conmemx,1,Len(Conmemx)-1)
If Len(Conmemx) <2
Exit && Exits if last character is
Endif && indicator or delimiter
Endif

Bx = Ltrim(Str(P_2))

If P_2 >7
Clear
@12,12 Say 'Irrecoverable Error on Record Structure,
Aborted'
Wait
Clear
Endif

If P_2 >0 .and. P_2 < 8
Do Case
Case Bx = '1'
E1 = D_8 && Main Entry Heading
Case Bx = '2'
E2 = D_8 && Uniform Title
Case Bx = '3'
E3 = D_8 && Title
Case Bx = '4'
E4 = D_8 && Edition
Case Bx = '5'
E5 = D_8 && Physical Description
Case Bx = '6'
E6 = D_8 && Dates
Case Bx = '7'
E7 = D_8 && Notes
Endcase
P_2 = P_2 + 1
Endif

If Lword = 'T'
Exit && Exits when last subfield processed
Endif
Enddo
RETURN

Fig 6.16 Subroutine Uconmem

314
Procedure Uconnote
** Unstrings Connotex into respective Notes, indicators
** ranging from a to x.
Anyflag = .F.  && Used to keep track of errors on
Loopcnt = 30  && indicators
Private D_8
Private P_1
Private P_2
Private P_2X
Private H_1
Lword = 'F'
P_1 = 1
P_2 = 1
H_1 = "-"

*** initialises Variables To Hold Notes
P2 = 65
Do While P2 > 64 .and. P2 < 89
    Ax = Ltrim(Str(P2)).
    Sx&Ax = Space(1)
    P2 = P2 + 1
Enddo

** Now Filling Each Note Using Chr(181) as Delimiter
** Leading delimiter removed in second line
If Len(Connotex) > 1 .and. Substr(Connotex,1,1) =
    Chr(181)
    Connotex = Substr(Connotex,2,Len(Connotex)-1)
Endif

** Tests for existence of Chr(181) and fills D_8 up to next delimiter
Do While P_2 <= Loopcnt .And. Len(Trim(Connotex)) > 0
    Connotex = Ltrim(Rtrim(Connotex))
    If Chr(181) $Connotex
        D_8 = Substr(Connotex,1,At(Chr(181), Connotex)-1)
        If Len(D_8) + 1 = Len(Connotex)
            Lword = 'T'  && Tests if last word being processed
            Else
                Connotex = Substr(Connotex, At(Chr(181), Connotex) +;
                1, Len(Connotex)- At(Chr(181), Connotex)-1)
            Endif
        Else
            Lword = 'T'
            D_8 = Substr(Connotex,1,len(Connotex)-1)
            If Len(Connotex) < 2  && Field must be at least 2
                && characters
                Exit
            && One for delimiter other for
Endif

P2X = Substr(D_8,1,1)  && Indicator is extracted
Ax = Ltrim(Str(Asc(P2x)))
If Asc(P2x) >64 .and. Asc(P2x) < 89
   Sx&Ax = Substr(D_8,2,Len(D_8)-1) && Variable filled
       & with contents of D_8
Else
   If Anyflag
      Clear
      @12,15 Say 'Too Many errors Generated from incorrect
Indicators, Process Aborted'
      wait
      Exit
   Endif
   Ax = '87'
   Sx&Ax = Substr(D_8,2,Len(D_8)-1) && Assigned
miscellaneous category when valid indicator not found
   Anyflag = .T. && Variable with indicator 'W' filled
Endif

P_2 = P_2 + 1

If Lword = 'T'
   Exit && Forces exit when last note processed
Endif
Enddo
RETURN

Fig 6.17 Subroutine Uconnote

All files and indexes have thus far, been defined. The
Dbase files were created using the standard Dbase
utilities. Indexes have been generated using the special
algorithms devised thus far. Data for the Depot file have
been entered using the option to create a code in the
input and update menus. Now that the E-R models have been
translated into appropriate files, the next phase of the

316
design process is to create input, update and deletion programs to produce the catalogue entries as stipulated by NAREM and AACR2R.

NOTES
1. At the conceptual level the structure of the date area corresponds to that of the NAREM dates. At the implementation level, however marked differences are observed as noted under section 6.5 of this chapter.
2. No such note is stipulated in AACR2R, but the note is essential for the sake of completeness when describing analytics. As NAREM reports consist of only collections, the note is not used.
3. This note is used to record the extent of item (for example, 23 leaves) and dimension for NAREM.
4. Usually the Reference number is of variable length. A fixed length field of 20 characters is used for the alphabetic and numeric portions which are assigned 10 characters each.
5. An example of excessive decomposition is given in chapter 3, section 3.4.1.3 of this thesis.
6. All programs form a tree, one leading to another. At the topmost or highest level, an application program calls all other programs beneath it.

317
CHAPTER 7
PROGRAMMING AND REPORT GENERATION

7.1 Introduction
The creation and maintenance of bibliographic records remains one of the most important professional duties in libraries and information centres. The efficiency and effectiveness of a service depends on these activities. Hudson identifies the following major components of a cataloguing system: addition, update and withdrawal of records from the catalogue, identification and correction of errors and application of authority control (Azbuike, 1988:276).

Automating the library catalogue today is common practice. Numerous advantages accrue from computerising a catalogue. These include instant addition, update and deletion; authority control over names, titles and main entry and retrieval based on one or more bibliographic elements. Different kinds of reports may be produced with varying amount of detail to suit the needs of individual users (Taylor, 1988:3). The fundamental logic of library automation is that the same bibliographic record, once in
its machine-readable form is capable of being utilised over and over again for different purposes for which a bibliographic record is required (Heseltine, 1990:163).

A computerised system should be easy to use (user-friendly), efficient, speedy and relieve users of unpleasant chores. In addition, the system should be flexible enough to deal with different kinds of users and provide alternatives for different purposes. It is agreed that using computer technology poses the problem not so much whether a solution exists, but what form it should take (Azbuike, 1988:280). Liskin (1987:24) mentions the following general objectives of any computerised system:

- to carry out all of the required operations and produce all of the requested reports.
- to optimise the performance of the system from the operators point of view.
- to maximise the accuracy of the information in the database.
- to facilitate maintenance, modification and expansion of the system.
When developing a library cataloguing system, these aspects must be considered. The system must be easy to use yet sophisticated enough to handle a diversity of applications and procedures. System's design begins with specifications of what the new system is to achieve and ends with a detailed system specification from which the programs are prepared. The ultimate system must achieve the goals set for it and reflect the structure of the business application it serves (Daniels and Yeates, 1984a:129). The entire system should be secure, accurate, reliable and easy to understand. The design process is not a straightforward process of starting with the system's requirements and ending in the design of input and output programs. It is instead an iterative activity where elements are examined and re-examined until every aspect of the proposed system has been considered and a final design has been reached (Daniels and Yeates, 1984a:130).

Application design cannot be divorced from database design because the content of a database is determined by what the users put into it and what they expect to get out of it. Application design and database design must proceed side by side, but, are handled separately for the sake of convenience (Rowley, 1990:92). When designing a
system, decisions as to what programs must be written, the complexity of each program, how long it would take to develop the programs and the method to test the programs are considered (Condon, 1982:229). However, programming cannot begin before the input and output specifications are clearly defined.

7.2 Requirements Definition
The requirements of a system is the result of demands placed on the system from all sources, for example, users and outside organisations. The input and output requirements arise from these demands. The output of a system consists of reports and records prepared to satisfy the demands placed against it. The inputs are used as a source to generate the necessary outputs needed to satisfy system requirements (Chapman, Pierre and Lubans, 1970:32-33).

The structure and format of a catalogue should be chosen with an eye to its function. All catalogues give access to a library collection via a number of access points. The average user uses the catalogue as a finding list, as an aid to discover whether a library has an item and where it is located on the shelf (Rowley, 1980:116-117). The composite model developed in the previous chapter of
this thesis is based on the AACR2R and NAREM requirements. Basic requirements of the AACR2R are reflected in reports that conform to these standards. The reports include the main entry heading (if any) and the ISBD. Examples of AACR2R reports were given in chapter 4 while examples of NAREM entries were given in chapter 5. Based on the requirements and reports required, the tasks assigned to the present system were determined and defined. This task was accomplished by processing the information at hand using suitable computer programs. However, before a discussion of how the tasks were accomplished using Dbase programs, some general design considerations are examined and established. Three aspects deserve mention, the validation procedures, screen design and program structure.

7.3 Validation and Control
Information must be valid when it is input, processed or when it appears on a report. For this to happen, controls over the information system at each step is essential. Strict controls maintain consistency and integrity of the database. A program is most adept at determining whether or not a precise set of conditions obtains (Malinconico, 1977:316). However, to achieve adequate controls, the rules themselves need to be precisely defined.
Controls are classified into input, program or output categories. During input, data can be keyed in by two different operators and the inputs checked. If the inputs match, they are accepted, otherwise corrections are made and then only are the inputs accepted. Another method is to compare the input data with the data in the source documents using online screens or hardcopy printouts. Input control totals and record counts are other methods to ensure that the totals tally and that all data has been accounted for (Ostle, 1985:181-183).

It is equally important to ensure that the input value is an expected value for that field. For instance, a Depot code must be a valid Depot code that exists in the system. The operator is presented with a table of valid input codes and is expected to select one of the alternatives (Ostle, 1985:186). However, systems often create more than one option. For instance, when a main entry heading is entered, the system tries to locate a similar heading already in the database and present this for selection. If it is still decided that a new heading is warranted, the user is allowed to enter the new heading as a valid heading.
While input controls increase the possibility that valid data was processed, output controls, ensure that output data are valid. Use of record counts, exception controls for example, excessive number of entries or transactions, larger than normal expenses or sales, are some techniques used. Controls detect undesirable trends or deviations from what is normally expected (Ostle, 1985:187-188). A dictionary, screen or program can test for values within a reasonable range.

Program controls are much more flexible than controls instituted by the dictionary or the screen. The program can ascertain whether the data is of a specific type (for example, numeric, date, character), of a specific length, whether it is blank, duplicated or whether a value is within a specified range, for example, for males and females, the system can be programmed to accept only two values 'M' or 'F' only. Range checks can ascertain that a certain limit is not exceeded, for example, credit limit not exceeding 10,000.00 rands. Additional checks can detect inconsistencies, for example, date of employment is greater than a persons date of birth. Programs can go even further by checking that two logically related
fields have the same values during editing (Ostle, 1985:184-185). When desired special routines may be developed to perform a special function\(^1\).

Validation procedures are required at each stage of a dialogue and data should be checked for completeness, format, range, reasonableness, consistency and sequence. When checking for completeness, a field with a fixed number of characters may require that there are no leading spaces or trailing blanks. Reasonableness may entail a check to see that the date of employment is not greater than the employees date of birth. Consistency can be checked, for example, by verifying data from more than one source, for example, online on screen and source documents. A sequence check can ascertain if the numbers are in increasing order and are not duplicated.

It is not possible to enumerate all controls used in the study. Hence a few examples of different types employed in the study are provided. These included limiting the input to specific case and type. For example, the classification number was defined as alphanumerics and was limited to upper case only. The control of duplicate records in the system was achieved by first making a test for the highest BRN in the system and this was
incremented by 1. This ensured that it was unique. The use of authoritative headings for the main entry to validate new data against existing forms served as a means of authority control. Evidence of validation procedures for the main entry headings, titles and depot codes are to be found in the input program, Iprog, discussed below. All authority file validation procedures follow a similar pattern. Hence, only the procedure to validate the depot authority codes is explained here.

The Depot authority file consists of accepted depot codes. During input or update, the operator is alerted when a code that does not match any in the authority file, is entered. By typing a '?' in the field reserved for the code, a set of acceptable choices is presented to the operator who may then choose one of the options or define a new code. A prompt for a description is provided in the latter case. Using this technique reduces the number of errors during input and update. The authority file is built over time. Fig 7.1 shows a part of the Depot authority file.
7.4 Screen Design

The screen is the most important interface with the user and controls the input and output of data. A typical screen is shown in Fig 7.2. The screen accepts a title, authors, edition, imprint, collation, series, subject heading, classification and standard number. A typical screen is divided into input and output message areas. Each field is identified by a label and is restricted in length and type. A program setting out a formatted screen on a VDU, is today’s ideal. Each field to be entered is either prompted for or appears as blank or default values on the screen, which the operator may overwrite.
Fig 7.2 Fixed Length Field Screen Format

A fixed format screen as illustrated above is problematic for bibliographic records as data does not always fit into such neatly formatted screens. In Fig 7.2, the series, for instance, may exceed one line and the length of large notes and abstracts are not predictable. In such
cases a 'variable format screen' is desired that can allow sufficient flexibility. To achieve this the technique of relative addressing is used whereby the @Say... get commands are placed in no fixed position on the screen but is relative to existing data on the screen. This is achieved by keeping track of all fields on the screen and using a counter to increment the rows and columns, for example:

@rw, 10 Say 'Author : ' + Author
@rw+2, 10 say 'Order date : ' + Dtoc(Lastorder)

where the variable rw keeps track of the current row, the value of rw ranging from 0 to 22. A typical screen has 25 lines and 80 columns numbered from 0 to 24 and 0 to 79 respectively. By using the row and column co-ordinates, the text can be positioned anywhere and data entered anywhere on the screen.

In this study, a uniform technique was used to display data on the screen. The screen was divided into a header and message area (top and bottom respectively) while the centre was used to display and accept data. The screen layout is illustrated in Fig 7.3. The header consists of a screen name and a heading for the screen. The message

329
area occupies lines 21-23 and alerts the operator of any errors, providing warnings and messages. The input/output area occupies the rest of the screen and is flexible enough to accept both structured and unstructured text of fixed and variable lengths.

The format and layout of the screen is controlled by the procedure 'Skeleton' found in procedure file Procfl. Three parameters are passed to the procedure, the heading, an instruction to clear or not to clear the screen and the screen name, respectively. A typical command such as "Do Skeleton with 'INPUT MENU,' 'Y', 'IMENU SCREEN'" sets the heading to 'INPUT MENU', deletes the screen (second parameter is 'Y'), and sets the screen name at position 0,0 on the screen.

```
SCREEN NAME
******************************************************************************
* HEADING HEADING AREA *
******************************************************************************

INPUT/OUTPUT AREA

******************************************************************************

MESSAGE AREA

Fig 7.3 Screen Layout
```
Input or output between the user and computer is essentially a dialogue. Daniels and Yeates (1984a:65) provide some guidelines for the design of screen dialogues:

- They must be easy to learn and use;
- Guidance to the user must be available on the screen;
- The dialogue should use the user's terminology;
- Response times should be as expected and the user informed of any delays.

Screens are forms to be filled out (Rowley, 1990:113). They are dynamic in that data is accepted, corrected and validated online; this may be accompanied by help texts to assist the operator in making choices. However, before a screen is designed, the sequence of processing and type of information to appear on the screen has to be ascertained first (Ostle, 1985:397).

7.5 Approach to Programming

Application programs are written to meet the processing requirements of a system's users. Every task performed by a computer requires a program or appropriate set of commands. The function of the program is to assimilate
and organise data, keep it up to date and provide selected information and reports from the database (Condon, 1982:69).

Good programming techniques lead to sound, well-conceived solutions. The top-down approach to problem solving breaks down a task into smaller manageable units until each unit has only a single function to perform (Orilia, 1986:205). In programming, the top down design can be employed by breaking down large tasks into modules that perform a specified task. All processing steps are assigned to their respective modules, the overall objective is to create a processing flow where a logical series of modules are executed (Orilia, 1986:228). Structured design assists in avoiding illogical or partial solutions to problems. Usually there is one main control module; control is thereafter passed to independent modules to which there is only one entry and exit point. Control returns to the module or command immediately below the preceding module or command in the main module. Using the above technique, a set of programs were developed to input, update, delete and generate reports for this study.
The topmost menu, called Aprog, sets all environmental details such as 'Set bell off' and 'Set talk off'. The Input program, called Iprog, Update program, called Uprog and deletion program, called Dprog and the report programs Nrep (for Narem reports) and Arep (for AACR2R reports) are options within the topmost menu.

An option to reindex corrupt indexes for Table1 and Depot files is provided as option 6 on the menu. Of particular importance is option three of the reindexing menu, where the headings are sorted in alphabetic order according to the five different levels outlined in chapter 6. Option 7 packs the files which works in conjunction with the delete option. Using this option all unwanted records are removed.

Finally, option 8 contains different utilities for the Depot record. These include an option to create, update the code and description, delete and retrieve the depot authority records. In addition, an option to exit the topmost menu is provided. These options are illustrated in Fig 7.4 below. Each of the above options leads through a series of submenus that are assigned more specific tasks.
Full comments are given in the program while only aspects of the program are discussed in this thesis. The topmost menu, input, update, deletion, report generation programs, reindex, packing and depot authority programs are found in Appendix C. All procedures called from these programs are arranged alphabetically in the procedure file, Procfl which is to be found in Appendix D.

**Fig 7.4 Aproq Options**

7.5.1 Input Program

Data input is the stage during which most errors enter the system and attention must be paid to minimise errors (Rowley, 1990:96). Input means information that is a matter of record within a system (Chapman, Pierre and
Lubans, 1970:65). When designing input programs, the designer must have three objectives in mind; the method of input must be cost-effective; the program must achieve a high level of accuracy, and the input must be acceptable to the user (Daniels and Yeates, 1984a:67).

The following are some considerations when designing input programs:

- the flexibility of the format depending on whether fixed or variable length fields are required;
- The speed with which the data needs to be captured, whether online or in batch;
- The accuracy of the data being captured and the extent to which it faithfully represents the original data;
- The method that is used to correct any errors (for example, immediate string replacement of the incorrect characters);
- Standardisation of the coding system and screen layouts (Daniels and Yeates, 1984a:76);
- Where and how the input originates;
- At what level the input enters or leaves the system, and the kinds of information appropriate at each level (Chapman, Pierre and Lubans, 1970:70-71);
- The sequence of the input data (This may involve grouping related options together) (Rowley, 1990:105).

Steps in Data Capture
A computerised system must not be designed independent of any clerical or manual system. The computerised input system is constrained by the latter (Daniels and Yeates, 1984a:71). Both the sequence and format of the dialogue must be acceptable and must follow the flow of actual transactions as it happens. In this study, separate sequences and utilities were not designed for the AACR2R and NAREM requirements. Instead the requirements were integrated into one logical sequence using a series of integrated screens.

The following steps outline the basic structure of the input program:
1. Set environment details
2. Open data files and related indexes
3. Prompt for different fields
4. Perform some operation on the field(s)
4. Store them
5. Repeat the process or exit the input menu
The AACR2R was used as the starting point for data capture. The order of the typical AACR2R entry could not be slavishly followed, however. The fields were prioritised and grouped logically. The following steps comprise the input process:

A. Input title, responsibility and filing article for the title. Verify the title, whether duplicate or not and decide whether to accept the entry.
B. Enter main entry heading and verify heading.
C. Enter the edition, dates, physical details, reference number, depot code, type of document, NAREM status and classification number.
D. Add notes and abstract.
E. Accept the new BRN assigned for the current entry.
F. Store item in database and continue entry of next new item, that is, repeat steps A to E.

These steps were covered by four basic screens for the title, main entry, notes and a screen for miscellaneous details such as the edition, collation etc. Intermediate screens for the verification of the depot, title and main entry were provided at sub-levels. Screen names used were mnemonic in nature, for example 'NOTES SCREEN' for the notes and 'UNIFORM SCREEN' for the uniform title option.
It was found not essential to test each field as it was entered. Instead all data were subjected to a battery of tests after each screen was filled. Any errors were reported in the message area. Using this technique enabled speedier data capture.

A. Title and Statement of Responsibility and Filing Article (Title Screen)

The International Standard Bibliographic Number (ISBN) is the ideal candidate for ascertaining any duplicate entries as it is unique. In the absence of the ISBN, the title is the next best option as it is the next unique property of an item. The edition, dates, physical details and notes do not qualify as they lack uniqueness. The main entry heading is much more reliable than the above properties, but is not an intrinsic property of an item only and applies to more than one item (works).

The operator is prompted for a title, a statement of responsibility and a filing article in the first input screen (Title Screen). Up to 500 characters was allocated for a title run under Clipper while 254 characters were assigned when run under Dbase. The commands, shown below, allow for larger titles to be used when run under
Clipper. The construct 'If Clipper ... else .... Endif' enabled usage of features in Clipper that were not available in Dbase. With the variable Clipper set to false, the program executes only the 'else' segment of the construct. Since Dbase is interpretative, the 'If' segment of the program is not inspected and therefore non-Dbase commands in this segment do not generate any errors.

If Clipper
Titlex = Titlex + Space(500-Len(Titlex))
Else
Titlex = Titlex + Space(254-Len(Titlex))
Endif

All incoming titles are checked for duplicates in the system. The program brings up selected items that have matching title keys (Tkey) and the operator is presented with a title, statement of responsibility, BRN and physical details for the suspected entry. When checking for duplicate entries the entire description is a valuable aid. Neither the title nor the main entry alone is sufficient. To properly evaluate whether the item being catalogued is a duplicate or not the entire description aught to be examined. Based on the

339
description, the operator decides whether to continue data capture or regard the entry as a duplicate in which case the input program is exited. All details on this screen are entered in upper and lower case. When a '?' is entered in the title or filing article field, appropriate help messages are received.

B. Main Entry Heading (Heading Screen)
The Main entry heading screen (Fig 7.5a) shows seven possible options. These include the names, place, corporate bodies, uniform titles, title, no main entry and exit options, respectively. An indicator on the screen shows the type of heading and the associated symbols. The status of the headings selected is indicated on the screen. For example, if the place and uniform titles were selected, this would be indicated by 'PU' on the screen. The program prohibits the entry of a title if either a name, place or corporate body is chosen. The name, corporate body and place are however mutually exclusive. The operator must clear the options already selected by selecting option 6 which initialises the headings and then the desired headings may be entered. Procedure main controls the conflict between the headings, and gives appropriate messages and suggestions. For example if a name has been selected as the heading,
and a request is made to add a place, a message 'Invalid Option chosen, Select Option 2 or Option 6 to clear existing headings' appears on the screen. Option 6 clears the headings thereby performing a deletion function. Each of the headings are controlled by the procedures, name, place, corp and uniform which are called from the procedure, main. The data elements captured for each type are shown in Figs 7.5b, 7.5c, 7.5d and 7.5e respectively.

**Fig 7.5a Main Entry Heading Screen Options**

The uniform title may appear on its own or together with one of the above type of headings (This has been discussed in chapter 6 of this thesis). Therefore it makes sense to request for firstly a name, corporate body or place and then a uniform title. Hence one of the three options are chosen first, before a uniform title is
chosen. Since the punctuation for the uniform titles may be a full stop, comma or a pair of brackets, the program prompts for one of these options (Fig 7.5e). The title on the other hand has been already entered at this stage.

When the title is chosen as the main entry heading, only the Utype field is updated and replaced with the value of 'T'. The values for Utype are U (if a uniform title) or T if a title, otherwise the field is blank. Similarly, the possible values for Mtype are A (if author), C (if corporate body), P (if a place) or blank if no main entry heading has been defined.

Entry Element :
Forenames :
Nobility :
Additions :
Relator code(s) :
British Term of Nobility (Y/N) :

Fig 7.5b Names

Place :
Additions :

Fig 7.5c Place

Corporate Body (including Subdivisions and parts where applicable) :
Additions :

Fig 7.5d Corporate Bodies

Uniform Title :
Additions :
Punctuation Separator -[Acceptable values are ".",""] :

Fig 7.5e Uniform Titles

342
The Procedures for the name, place and corporate bodies have a similar structure and program code. If the length of the first element is 0 or the program name from which it is called is 'UPROG', the variables El1 to El6 are initialised, otherwise procedure Uconmain is invoked to fill the variables with the content of the subfield Main. Thereafter the elements are presented for update and appropriate help and control checks are instituted. For example, for the names, only the values of 'Y' or 'N' are permitted for the field, British. The entry element is mandatory. Procedure Mainval checks for matching headings in the database and displays them on the screen. The headings are either accepted, skipped or the menu is quitted. The accepted heading may then be amended as desired.

Procedure Wcomp2 generates the key for the name heading. The key is used to locate matching headings during validation. Finally, all elements of the main entry heading are stored in the variable mainx using the procedure conmain which prepares the storage format for the heading.
C. General Screen (Utility Screen)

The Utility Screen (Fig 7.6) accepts the edition statement, dates, physical details, reference number, a depot code, NAREM status indicator, type of document and classification number. The NAREM status indicator is either yes or no ('Y' or 'N' values). The latter option applies when the entry is not to be submitted to NAREM. The type of document includes only accessions and manuscripts. Hence only two values 'A' and 'M' were accepted by the program. The edition is mandatory if applicable while the physical details are mandatory. The ISBD punctuations apply in both the fields. The reference number is mandatory if applicable since not all examples have reference numbers. The depot code is mandatory and is validated against the Depot codes in the Depot file (already discussed under section 7.3). The edition, physical details, dates and depot codes are entered in lower case while the reference number, Narem status indicator, type of document and classification numbers are entered in upper case. The program, however, automatically converted the characters in these fields to upper case.
**Fig 7.6 General Screen (Utility Screen)**

The Dbase date format is emulated on the screen and has the structure mm/dd/year for the Date1 and Date2 fields. The format is embedded in a 8 byte character field that accepts numeric data only. Each of the three segments are tested. The month and day were required to be in the ranges 0 to 12 and 0 to 31 respectively. Only the Month/year, Month/day/year options were allowed. The existence of the day and month without the year was not allowed. The Dates field, on the other hand, allowed any type of character. Hence it was able to accommodate any
expression representing the date, for example, 2 May 1978 or 1978? Due to its free format, the Dates field could not be adequately validated.

Between the title and date area (AACR2R), the date is mandatory. Its existence in the Date area is optional. Hence a control mechanism, using field Dopt, was used to override a blank date, whenever one was detected.

D. Notes (Notes Screen)
A wide variety of notes may be selected. Additionally, two miscellaneous notes not catered for by NAREM or the AACR2R were available. The summary was considered a note and included the ‘+’ character for NAREM purposes. The options for the notes are shown in Fig 7.7.

The notes utility includes a query facility to ascertain and view the notes included in an entry. The operator enters a ‘?’ instead of selecting an option from A to X. The Brownote procedure is invoked which lists only the notes that has data held in the current variables. Once all notes have been entered, the Connote procedure strings all the notes together. The resulting variable, connotex, is then passed to the Conmem procedure for storage as a subfield of field mem in Table1.
Fig 7.7 Notes Screen

Once all the data have been captured for an entry, a BRN was finally assigned to the current entry. This was followed by a battery of Replace commands which updated the fields in the record for the current entry. All associated indexes were simultaneously updated.

7.5.2 Update Program (Uprog)

The term update is very general and may be equated with the term file maintenance. Update involves modifying the master file with current information by adding new data.
or changing old and incorrect data. The primary objective of the update program is to keep the file in the most current state (Chapman, Pierre and Lubans, 1970:167). The update program is the second option in the topmost menu (Aprog). The following steps constitute the general outline of the program:

A. Accept/Input Valid BRN  
B. Display Record details  
C. Select Options to be Updated  
D. Exit Menu

UPDATE OPTION SCREEN           12/11/93
***********************************************************************
*                      UPDATE PROGRAM                           *
***********************************************************************

A. Title and Statement of Responsibility
B. Main Entry Headings
C. Other Fields
D. Notes
E. Exit Update

Select Option : A

***********************************************************************

** Fig 7.8 Update Options **

348
The options in the update program are shown in Fig 7.8. Each of the elements, except for the BRN that was available for input in the input program, are available for update. The program uses the same utilities as the ones used in the input program and therefore a discussion of them is avoided here.

To update a record, access is gained via the BRN. This is the first field that is prompted for (Update Screen). Thereafter a full screen display (Update View Screen) of the record is presented for viewing and update (Update Options Screen). Appropriate fields based on the option chosen are presented for update. Note that the Tkey, Mkey, Utype or Ntype are not presented for update but are automatically updated each time a title or main entry heading is updated.

There are a few significant differences in the update program when compared to the input program. Firstly, the record must exist in the system to be updated. If it exists the details of the entry are passed to different variables which are presented for update, for example, Depotx, Refx, Classx etc. Where a field has not been entered, it is usually blank and is nevertheless presented for update which is then subject to the same
validation rules as in the input program, except for a few changes. In the update menu, title matches are not made as the title has already been accepted. It is unlikely that the main entry would change and is therefore not subject to validation.

7.5.3 Deletion and Packing

When the delete program (Dprog) is invoked, the BRN of the entry to be deleted is requested. Thereafter, the operator is requested to confirm the deletion of the record. The record in Table1 is then marked for deletion and all deleted records and index entries are removed when Table1 is packed using option 7 of the topmenu.

Sometimes an item may be have been wrongly deleted or requires to be reinstated. The delete program, firstly ascertains whether an item has already been deleted. If the item has been deleted, the operator is requested to recall the item or leave it in its present deleted state, otherwise the item is deleted. The program statements to effect this are shown below. Before the program is exited, however, the operator is reminded to pack and reindex the files to tidy the database.

    If Deleted()
        Dopt2 = 'Y'
        @11,5 Say 'Item already deleted'
7.6 Reports

Outputs are information transmitted by the system (Chapman, Pierre and Lubans, 1970:65). A typical library catalogue identifies different data elements, inter alia, the main entry, title, edition, imprint, physical details and notes. Printouts based on these elements must be fashioned for various needs and be in a form convenient to the user (Condon, 1982:274-275). In some cases the internal storage format require reformatting, for example, conversion of a date from dd/mm/yr to mm/dd/yr or into equivalent months days and year, for example, 17 May 1953. The flexibility of a DBMS is put to the test when producing a variety of output formats. By generating the desired reports, it confirms whether the database has adequately catered for data in its files (Liskin, 1987:47).
The structure and presentation of the reports have to be specified in advance in any system. Standards are essential when designing reports. The size of the report, type of paper to be used, spacing, margins, typeface, headings and number of lines needed are important factors (Condon, 1982:268). Equally important is the appearance, spaciousness and order (logical sequence) in which the data appears in the report (Rowley, 1990:117). The content of the report is however, most important. Hence precision is vital when designing reports.

When generating and designing reports, the following questions serve as a guideline:

- How often will the report be printed?
- Who will use the report?
- What specific items of information should be included?
- For a columnar report, in what order should the columns be printed?
- For a full-page report, where on the page should each item be placed?
- In what order should the data records be printed?
- What calculations must be done within each record?
- What subtotals, totals and statistics are required?

The easiest way to obtain answers to these question is to obtain a rough sketch or template of each page of each new report with some representative data (Liskin, 1987:13).

The variable length records for bibliographic material makes output difficult using Dbase. The position of the field does not remain fixed in a variable format. Also each new field starts on a new line and this is not desirable. A command such as List Author, Title, BRN and Class is not intelligible to the system as the memo field was subdivided into fields which are not accessible to the Dbase reporting commands. Hence special routines were developed to produce the reports for NAREM and AACR2R. Two such routines, Nrep for NAREM reports and Arep, for AACR2R reports have been developed. They governed the output format through appropriate language statements.
In both types of output, the format of the entries were relative. Hence, no pre-printed output forms were used. All labels and data were inserted simultaneously into appropriate areas of the report. Each entry began on a new page.

7.6.1 NAREM Format
Institutions holding manuscripts collections in South Africa are required to submit their records to the national register (NAREM). Participants to the project are required to enter information on a standard worksheet (Fig 7.9) providing details such as the depot, type, accession number, dates, title, remarks and summary for each entry. Automation opens new opportunities in the way data is submitted to the register using the worksheet as a guideline. Yet it immediately poses questions of standards. How will the data be communicated and in what format. The order and elements to be included in the report has been already specified by NAREM (Fig 7.9) and this served as a useful guideline during program and report design.
Fig 7.9 Worksheet Format

The elements for the NAREM reports were extracted from the composite model and a worksheet was reconstructed from the data available. The utility, Wrep, positioned each paragraph appropriately, based on the length of the field being handled and the positions of the previous and
successive paragraphs. The title, remarks and summary paragraphs were of variable length and therefore occupied more than one line in a report. The remaining paragraphs were allowed one line each as they were not expected to exceed one line. Each entry began on a new page while each new paragraph commenced on a new line. The abstract, due to its unpredictable length sometimes occupied more than one page.

Only records reflecting collections that were flagged in the database (the field NAREM = .T.) were included in the report. File, item and lower level records were excluded. The reference, depot code and type were extracted from the database which exist as fields. In the case of the latter only a one letter code was stored having values 'M' or 'A' for 'Manuscript' and 'Accession', respectively. The code was 'exploded' into its fuller form when the report was generated. Fig 7.10a illustrates an archive entry with type 'Accessions' while Fig 7.10c illustrates an entry of type 'Manuscript'. The dates were extracted from the Date1 and Date2 fields. Example 7.10a illustrates the opening and closing years only. Where the day or month did not exist, they were omitted from the opening and closing dates. The title and abstract were taken from the title and note fields, respectively. The
remarks paragraph was constructed from different notes including notes on the donor, placement, finding aids and access restrictions. The extent of item also formed part of the remarks paragraph. Fig 7.10b illustrates a donor note and a note on the availability of an inventory. Example 7.10a illustrates the extent of item in linear meters and contains a donor note as well. The abstract in Fig 7.10a contains '+' characters to reflect the series and arrangement of the collection.

Three selection criteria were used when generating the report. These were the Depot Code, the category of reference (for example, references starting with 'A') and BRN range. The reference allowed for different reports to be produced for different categories of materials while the BRN range allowed selection on one or more records.

The following examples illustrate different kinds of reports. Fig 7.10a illustrates an entry for an archive depot (accessions), Fig 7.10b, an entry for manuscripts taken from the Unisa Documentation Centre. Fig 7.10c shows a collection level entry taken from the Bhana collection (UDW). Details such as the classification

357
numbers and main entry heading were not included in the entries but were included in the AACR2R entries below (compare Figs 7.10c and 7.11a).

**DATA FORM**

Depot : KAB
Type : Accession
Reference : A920
Title : Mrs. BBA Bridger
Starting Date : 1892
Ending Date : 1951
Remarks : Donor : Mrs. BBA Bridger, 1959: 0,35m (4 v.)
Summary : Helena Johanna Francina Lochner (1892-1951), Afrikaans authoress and illustrator of stories for children.

+ Correspondence with publishers, 1948; manuscripts of stories and articles published by Lochner; scrapbook containing press cuttings and advertisements of her publications; sketch-books; autobiographical notes; genealogy of the Lochner family; photographs.


**Fig 7.10a Archive Depot (KAB)**

**DATA FORM**

Depot : Unisa
Type : Accession
Reference : AAS8
Title : MG Buthelezi Collection
Starting Date : 1970
Ending Date : 1983
Remarks : Donor : MG Buthelezi; 0,80m; Inventory.
Summary :
+ Speeches by Mangosuthu Gatsha Buthelezi, Chief Minister of KwaZulu, on a wide variety of topics.

+ Press releases.

**Fig 7.10b NAREM Entry (Unisa)**
DATA FORM

Depot : UDW
Type : Manuscript
Reference : 001281
Title : Bhana collection / Surendra Bhana
Starting Date : 1860
Ending Date : 1987
Remarks : 1 metre
Summary : Collection consists of scrapbooks, newspaper cuttings, photographs and documents on history, politics, indentured Indians, Protector of Indian Immigrants, University of Durban-Westville and the Indian Trader.

Fig 7.10c Bhana Collection (UDW)

7.6.2 AACR2R Format

The basic components of an AACR2R entry is the main entry heading (optional) followed by the ISBD. A typical AACR2R entry for manuscripts consists of 5 areas, each separated by a full stop space dash space ‘.-’.

Indentions were introduced at the title area and for each of the notes in the note area. When constructing an entry, the main entry heading, title, edition, dates, physical details and notes were taken from the fields as defined in Table1. Only the abstract posed a problem. The program searched for the ‘+’ characters appearing at the beginning of each line and removed them from the summary. Typical entries generated from the system are illustrated in Fig 7.11a to Fig 7.11g. Fig 7.11a illustrates a collection (Bhana collection) with a name main entry

359
heading followed by the ISBD while Fig 7.11b illustrates an entry with no main entry heading being defined. However, a typical AACR2R entry is not complete for the purposes of identification and location. Hence entries all entries were preceded by a BRN, reference, source (substitute for Depot) and classification.

**Fig 7.11a AACR2R Entry with Main Entry Heading**

**Fig 7.11b AACR2R Entry without Main Entry Heading**

360
Lawyer and government official. Correspondence, memoranda, orders, reports, official and unofficial policy papers, draft legislation, drafts of writings, clippings, and printed matter, chiefly 1945 - 1948. The papers relate almost entirely to Russey's work with the Government Section, Supreme Commander Allied Powers (SCAP) during the Allied occupation of Japan following World War II and the efforts of the Allies to reorganize Japanese society and government. The files contain detailed information on nearly every aspect of postwar Japan, especially the drafting of the Japanese Constitution, the reorganization of the Diet, and policy proposals in the areas of civil rights, prisons, the economy, prostitution, and the imperial household. Correspondents include George Atcheson, Jr., Roger Baldwin, McGeorge Bundy, James F. Byrnes, Charles L. Kades, Douglas MacArthur, Wilo Rowell, Ray Sakakibara, Robert Ward, and Courtney Whitney, as well as many officials of the Japanese government. Also included are some personal papers and correspondence of Russey relating to his interests in such areas as civil rights, the CIA, and his various writings.

**Fig 7.11c Entry from Hensen**

Speeches by Mangosuthu Gatshe Buthelezi, Chief Minister of KwaZulu, on a wide variety of topics.

**Fig 7.11d NAREM Entry in AACR2R Format**
SUMMARY:

Volume includes the ff. documents:

Internal developments / NIC. re: UWO, unemployment, education, Non Racial Congress, housing and amenities.

Memo... SAIC to United Nations (1960). re: State of emergency

Agenda book (23rd Conference, 1961) / SAIC. re: apartheid
Indian Affairs Department

Natal Indian Congress - constitution

Memo presented to International Cricket Conference (1974) / SA Cricket Board of Control. re: apartheid in sport

The Congress position on the new Constitutional proposals. re: House of Delegates, Presidents Council, franchise


The White Fear / Conference on Race Discrimination

Inter Cabinet Council

Statements by NIC. re: Fatima Meer, elections, detente, SAIC, Oriental Bazaar Traders

The negative influence of cultural and economic disengagement from South Africa on intercommunity relations / Pat Poovallingham

Petition by SAIO to Parliament (1951). re: Group Areas Bill

Fig 7.11e File in Bhana Collection

362
The example in Fig 7.11c is taken from Hensen and illustrates a name heading, abstract, biographical note, finding aid and the source of purchase of the collection. Fig 7.11d illustrates a NAREM entry translated into an AACR2R entry. It contains a title, a date in the date area, extent of item, summary, notes on the donor and finding aids but lacks a main entry heading and classification number. Fig 7.11e shows a file within the Bhana collection (second level description) while Fig 7.11f illustrates an item within the file in the Bhana collection thereby exhibiting a three level hierarchy.
within the collection. Example 7.11f contains a relationship complexity note consisting of a title and accession number. Both examples in Fig 7.11e and Fig 7.11f have 'in' analytic notes and classification numbers. Finally, the example in Fig 7.11g illustrates an entry for a thesis which contains inter alia, a statement of responsibility, a physical description that includes the number of leaves, a miscellaneous note that indicates a copy located in the Indiana collection and another note indicating that a bibliography exists. The thesis note states the degree conferred, the name of the university conferring the degree and the year in which it was conferred.

The chapter has thus far examined the feasibility of producing NAREM and AACR2R reports from one composite model. Reports were subsequently produced from the composite model satisfying both requirements. These results have illustrated that both the formats are compatible, notwithstanding, a few constraints.

The '+' character had to be included in the abstract during data capture even though it was exclusively a NAREM requirement. To make it acceptable for AACR2R purposes it was removed during output. Separate dates had
to be defined to suit the needs of NAREM and the AACR2R. The AACR2R dates were accepted as is but the dates for NAREM entries required a fixed format and data conversion to blank the days and months when they were zero. Based on the reports generated from the sample data, the composite model has demonstrated that collection level entries for NAREM purposes and at least three levels of description for the AACR2R could be successfully produced.

The programs that were developed to capture and retrieve the data required modifications. The sample data could not be entered as is and required interpretation. The modifications made to the programs and situations requiring a shift in the design strategy, together with other findings are discussed in the next chapter. Thereafter recommendations for future areas of research are proposed.

NOTES
1. See overdue routine developed in chapter 2 of this study.
CHAPTER 8

FINDINGS, RECOMMENDATIONS AND CONCLUSION

The chapter documents numerous findings and observations based on the study. Special attention is paid to initial decisions taken in terms of modelling, programming and choice of sample, to mention just a few. A series of recommendations are subsequently made based on these decisions and they point the way to possible areas of future research. The study finally concludes with comments on the way forward for archive, library and information services in providing a more integrated and relevant cataloguing code.

8.1 Findings

General Findings

While attempts were made to adhere to the underlying principles and practices of the AACR2R, it was not possible to always translate entries into exact equivalents in the AACR2R format. For example, in the remarks paragraph, a description such as ‘20 items and 50 photographs’ for the physical description required reformatting into AACR2R equivalents. The objective of the study was to use the AACR2R as a basic model in order
to arrive at a NAREM compatible model arising from extensions to the basic AACR2R model. 'Two standards are fully compatible if records using either can be algorithmically transformed to the other, and back again, without any loss of information whatsoever. This level of compatibility is full reversibility' (Crawford, 1982:274). 'Compatibility' in the context of the present study was therefore interpreted as the ability of the composite model to reproduce data in the correct paragraphs, not necessarily as to the order of the elements in the source documents. Attempts have been made to keep the original order where possible. For instance, in the remarks paragraph, the order of the elements, namely, donor details, extent of item, placement of items with other collections, existence of finding aids and any access restrictions were kept intact. The precedence order of the APPM or AACR2R could not be kept however (see appropriate codes books for APPM and AACR2R). With respect to the latter (AACR2R), the introduction of 'in' analytics and relationship complexity notes upset the normal order of the notes. The model established its own precedence order as defined in chapter 6.
In the APPM the biographical note exists as a separate note. A typical NAREM and AACR2R summary, on the other hand, has a historical or biographical note which precedes the listing of any files and series. Hence, during data capture, it was resolved that the biographical or historical note would be included in the summary for all cases when a summary was required. This facilitated AACR2R and NAREM compatibility. Where no summary was required, the history or biographical notes were entered as independent notes.

With respect to the capture of the summary, the ‘+’ character was used for each file or series. The ‘+’ character did not pose any problem when translating the summary into an equivalent summary for the ISBD. However, the format of the data during data entry presented problems when entries were output for NAREM. Since up to 10 characters were assigned for the label ‘Summary : ’ in the report, the initial line had to be indented at least 10 spaces. Hence the program was modified to test for this condition failing which, printing began on the next line. This option accommodated those cases when it was desired that printing was to begin on the line following the summary. This option was exercised when the summary began with a listing of the files and series.
While the composite model defined the physical details to be mandatory for AACR2R entries, the actual data reflected otherwise. Many of the KCM entries did not have physical details. Some NAREM/archival entries (among entries 1-30) exhibited similar traits. Hence the program had to be modified to reflect this position. The status of the field was changed to mandatory if applicable. This is contrary to the AACR2R requirement that the physical details be mandatory. In a wider information environment, where no code for cataloguing exists or when the entries are incomplete, a change in the status of the elements is to be expected.

The sample displayed varying levels of completeness. Records belonging to NAREM or the Archives services always had a depot, type and reference paragraphs and these were found to be wanting in other records. A compromise was reached on the use and function of the depot code in AACR2R entries. The code was used to identify the Source, that is, a particular institution or publication the entries were taken from. While the inclusion of the source, reference number, classification and BRN is a departure from the standard AACR2R entry they serve a useful purpose in locating and identifying
collections and entries. In the case of the KCM, UDW and UDWT entries, the internal sequence number found in the sample were used as the reference number, thereby acting as a 'local' BRN or reference. Such numbering sequences were however missing in the case of the APPM entries.

The opening and closing dates for NAREM entries often required interpretation into equivalent date formats to make them compatible with AACR2R entries or fit them into a fixed date format, namely mm/dd/year. The format depended on the dates in the source documents which were at times not translatable into exact equivalents and were therefore translated into the nearest date(s), for example, '1976?' as an opening or closing date was set at '1976' but recorded as '1976?' in the date area to reflect the correct dates. Where the date was excluded from the title, attempt was made to include the exact or approximate date(s) to make the entries AACR2R compatible. No problems on the dates were experienced with the Documentation Centre (UDW) entries which were sensitive to both NAREM and AACR2R requirements.

The entries for the Killie Campbell Library (KCM) contained many responsibility statements while the others generally lacked such a statement. The practice
not to include a responsibility statement after the title is in keeping with manuscript and archival tradition. The main entry heading, instead, served as the 'provenance' or main responsibility for the work.

Finally, collections in the sources from which the sample data was extracted, tended to have both published and unpublished material. However, the study limited itself to only unpublished material in the selected collections. The Bews collection contained a published item which was omitted from the sample. Collections in the UDW source had large numbers of published material and were excluded from forming part of the sample.

E-R Modelling and Bibliographic Data
The basic E-R model was found to be inadequate to illustrate bibliographic elements, graphically. In the study, extensions to the basic E-R model were developed in order to map properties specific to bibliographic elements. For instance, three categories of elements, mandatory, optional and mandatory if applicable, were introduced into the model. Mandatory if applicable was specifically defined to relate mandatory elements with the existence property of an element. For a given level of description, an element may not exist but is mandatory.

371
if it does exist. This is a departure from the view that a mandatory element must exist. Instead, a weaker version of 'mandatory' was developed, that is, if an element was required at a specific level of description, the element was mandatory only if it existed. Mandatory elements, for example, a BRN for a record must exist, while optional elements may exist but are not necessarily included in the description. These extensions to the basic E-R model were found to be adequate.

The basic E-R model did not make provision for the different kinds of entities. In the study kernel, characteristic and designate entities were defined and identified. Both hierarchical and subordinate entities were defined in terms of whether entities were characteristic, kernel or designate. However, more semantics could have been developed in the study to map different properties, for example, the existence of a date in the date or title, or the existence of a statement of responsibility even if no mark was present to alert it's existence. While these findings point the way to some of the short falls in the E-R model used in the study, they did not affect the reports that were generated. For instance, an override option was placed in
the program to alert the operator whenever a date field (date area) was left blank. He could then override the option or enter a date in the field.

**Dbase and Bibliographic Elements**

Dbase is not adequately equipped to handle variable length records or repeating fields; even though, in the study, a memo field was used to accommodate variable length records and repeatable fields. Special algorithms had to be developed to identify and isolate fields and groups of fields. The Dbase dictionary proved to be inadequate in this regard. It is unable to accommodate subfield structures and identify them as independent elements. The failure to identify subfield structures have implications for generation of reports and indexing which normally rely on file and field definitions. Specialised routines (Nrep and Arep) had to be developed to control report generation for NAREM and AACR2R reports.

Dbase indexing techniques are limited, for example, indexing on upper/lower case or parts of a field including one or more fields. Dbase cannot index on keywords or isolate any filing article when constructing an index. Accordingly, the filing article was isolated
and assigned a separate field to facilitate indexing. In addition, specialised utilities were developed to construct the index entries. Where indexing was based on variable length fields and required computation, a special index was defined and constructed. The mainindex index is such an example. It came under direct program control (option 3 of the Reindex procedure). Neither the Dbase dictionary nor the standard indexing options proved to be adequate. The technique of indexing as explained in chapter 6 for main entry headings, using the mainindex index, proved to be adequate.

The design of archival systems require more than just identifying common elements shared with bibliographic elements. A substantial part of archival description consists of notes. These require the use of a word processor to insert and delete lines, words, indent the text, move blocks of texts around or move between screens (Roe, 1990:153). These facilities must form part of any new system envisaged. Dbase proved inadequate in this regard and the Clipper built-in editor was used instead to input and update abstracts. Formatting the abstracts using the editor proved expensive in terms of storage space as the formatting commands in the editor, utilised excessive space. Hence, a fixed length field of 800
characters was used for the remaining notes which utilised only the storage space as determined by the length of each note.

Finally, the replace command in Dbase could not be used to store the mem (memo field) but Clipper was able to achieve this with ease. New data was added easily. It would not have been practical to insert the delimeter commands in the mem field, as would have had to be the case if only Dbase was used. The fields, for example, the main entry heading, title, uniform title, physical details in the mem field, under the control of Clipper were updatable individually using the special routines conmem, uconmem, connote and uconnote to access the individual elements of the mem field. Since the replace command could not be utilised satisfactory in Dbase, usage of this particular feature meant that data could not be stored in the mem field when run under Dbase. However, with Clipper set to true (.T.) in the topmost menu, any Dbase clone that allows the use of the replace command for memo fields can be used, with due considerations to restrictions placed on the clone due to existing language statements in the program.
Screens and Reports In Dbase

Both the screen and report generators are not adequately equipped to handle variable length data in a dynamic way. A report generator should be able to remove unwanted trailing spaces and make the report more compact, so that the next field to be output appears in the next available space (not on the next available line). Where the next field appears after the maximum length of the field is output, much space is wasted. To overcome this problem relative addressing and grouping fields into convenient display or output units were utilised, for example combining the reference number, BRN, and source as one string and using the '?' to output the string as one unit of information.

Ideally, an operator should know which record is being added or updated. A uniform screen was subsequently devised to provide for messages, online help, headings and an input/output segment for uniform presentation to accommodate variable length data. However, when working with memo fields, the current working environment is not apparent to the operator. Clipper facilitated this by adding comments and a help text during the input and update processes. This is particularly noted in the
creation and modification of the abstracts. In other instances, the message area of the screen was used for messages and help text.

With respect to the AACR2R, the issue of screen design and the variety of output formats have not been considered sufficiently. Both the screen and output formats can be standardised. These can include standards in terms of terminology, screen prompts, output messages, error messages and an entire array of standards so that when using a standard 80x24/25 screen, the areas are defined and known, for example, lines 23 and 24 reserved for messages and line 1 to 3 for the header. In doing so, one form of communication for Information Retrieval systems would be used as a standard. Such standards should not be subject to the whims and fancy of each vendor. Librarians must lay down standards for vendor-based systems.

Hagler notes that 'output formats have gone somewhat adrift of the code and a truly integrated set of rules would incorporate, in addition to rules for description and access, also rules for the ordering and display of bibliographic records' (Svenonius, 1989:184).

377
NAREM

The NAREM sample data lacked important elements such as the edition and main entry heading to enable AACR2R compatibility. However, this was not the objective of the study. Instead institutions using the AACR2R and wanting to report to NAREM had to make provision for 'complete' NAREM entries as extensions to the basic AACR2R model (chapter 4). A degree of incompleteness was noted, however, when converting NAREM entries into AACR2R entries. In the alternate entry approach where the main entry heading was not required, the entries were more or less complete. Entries that did not have dates in the title were deemed to be incomplete in terms of the AACR2R. Hence a date was entered in the date area if not already in the title area to achieve compatibility. Most NAREM entries had titles, notes and physical descriptions. None had edition statements.

NAREM lacks a cataloguing code. A code would specify the order of the elements in the description, the order of the elements within the paragraphs, the conditions under which specific elements are to be entered and possibly facilitate machine translation if unique subfield identifiers are used. The lack of a code means that minimum standards may not exist or be adhered to. The
title, summary and dates are optional in NAREM entries. The remaining paragraphs, the depot, type, reference and remarks, when included, would lead to an entry that is meaningless. When converting such records to AACR2R, only the remarks and type would be relevant, which means that only the physical description, notes and GMD would contribute to an AACR2R entry and this is not acceptable as an AACR2R standard. A minimum standard is required for both NAREM and AACR2R entries to be 'reverse compatible' and acceptable. Level one ISBD exists as a minimum for the AACR2R, but no minimum standard exists for NAREM entries.

NAREM is not suited for machine translation at levels lower than the paragraph level. The lack of a well-defined internal structure for the remarks paragraph, for instance, makes it impossible to extract individual elements for an AACR2R entry. However, a structure was imposed by the composite model with regards to the order of the elements but a reverse translation would not be possible unless each element is identified uniquely in the paragraph. The ';' separator which was used in the paragraph is not adequate as a delimiter as it has grammatical connotations and confusion may arise over it's usage.
AACR2R and Manuscripts

Chapter 4 of the AACR2R is not adequate to handle the range of descriptive elements applicable to manuscripts. It falls short in many respects. The notes are not extensive enough to handle the diversity of situations warranting notes, for example, notes on a repository, biographical/historical notes or notes on the relationship complexity of an entry with other entries.

The code needs to be expanded beyond a two level description for analytics. Even though provision for multi-level description is made for materials to be described in one record, it's use is not common in libraries. Analytical entries at three levels is provide for in the code, but manuscripts (chapter 4) do not have a series statement and therefore 'in' analytics are limited to only two levels. The depth of analytic entries should go beyond two or three levels. In the study a relationship complexity note was used to link the item/piece to a collection or higher level of which it formed a part. Further, the part or segment of the collection being catalogued required highlighting and locating. These shortcomings have been identified and appropriate structures, namely, 'in' analytics and
relationship complexity notes were created and used to enable a more complete description of an item. In this regard the AACR2R is incomplete. NAREM, on the other hand, describe complete collections and the identification of the parts were not necessary.

The code lacks guidelines on how to sort and arrange manuscripts. A collection needs to be sorted and arranged before it is catalogued and described. A brief explanation of the principles and practices of the archival tradition (even in an appendix), should form part of the code and reliance should not be placed on some external source for such basic information. Just as order is demanded in the way entries are constructed, sufficient attention should be given to the internal structure of a collection for it to be properly described and catalogued.

**Individual Manuscripts and Type of Materials**

The study placed emphasis on the cataloguing of only manuscripts to the exclusion of other types of materials and this approach creates it's own problems. Unpublished material such as reports, agenda books, conference papers, audio-visual tapes, maps and music sheets that form part of a collection may require separate treatment
and there is a need to identify such categories which are covered in separate chapters of the AACR2R. The GMD and description of individual items may differ from that of the parent record. For example, a map or music sheet in a collection which when described separately has it's own GMD and physical description relevant to maps and music sheets. Further, manuscript collections are not necessarily fully described by the GMD of 'manuscript' but require an additional GMD, for example, manuscript on microfilm. In this case the 'microfilm' is the primary medium and is described in terms of chapter 11 of the AACR2R.

The model has excluded descriptions based on the type of material for special materials mentioned above. The difference between published and unpublished material is becoming smaller and the rules for manuscripts in chapter 4 of the AACR2R are far from adequate. The decision to include unpublished material in different chapters of the AACR2R, is a positive one, but considerable overlap exists between these categories of materials and the individual pieces catalogued in terms of chapter 4 of the code. Codification on 'manuscripts and unpublished' material require greater re-think and integration.
International Standard Bibliographic Description

While the fixed order of the elements and punctuations keep the business rules of the ISBD intact, DBMS cannot use the format as a database format in its present form for a variety of users. Different users may require different types of output. The ISBD is not machine-readable and is therefore inflexible. A more flexible format for database implementation consistent and compatible with the ISBD, is desired. The ISBD areas group related items together but this does not mean that they form part of a logical entity. For instance, the GMD applies to materials as a whole and does not logically belong to the title and statement of responsibility area and is poorly placed as a subfield within the area, even though being placed early in the description alerts the user of the nature of the material. A more comprehensive format with a system of tagging as in the MARC format, would facilitate a more flexible approach to element identification and output for varying user needs.

In the final analysis, both the AACR2R and the ISBD are incomplete models. For instance, not all authors are included in the description or as access points. The failure of the AACR2R to model the third level

383
descriptions, points to the inability of the model to accommodate variations adequately. For instance works by collective authors are not modelled adequately by the ISBD, as these exceptions are not catered for in a single model. The exceptions play havoc to the normal order of the elements. Nevertheless, the models serve as a useful point of departure for the analysis of the AACR2R and ISBD elements.

Neither the ISBD nor the NAREM format describe manuscript and archive collections adequately. Taking the notes, for instance, the sample data indicated whether an item was a gift or purchase or whether finding aids were available. Additional information such as the National Union Catalogue numbers (for example, NUCM MS 78-1683) or the archive group, for example, 'Forms part of: record Group 401' were also included in a note. Similarly, where a collection was described in a published source, for example, Guide to the National Archives of the United States, this was given in a note. In short, while the notes of the AACR2R and NAREM included some of the above aspects, neither made provision for all categories of notes. The options provided in the notes procedure were a
combination of both requirements but a complete format to include a wider variety of notes is required for a more complete model for manuscripts and archives.

8.2 Recommendations

The following recommendations are made in the hope of providing guidelines for future areas of research with respect to manuscripts, the AACR2R, NAREM, modelling techniques and published and unpublished materials.

- That the AACR2R place greater emphasis on explanations of the principles on which manuscripts and archives are based. These should include principles and guidelines on how to sort and arrange manuscripts either in the main body of the text or in an appendix.

- Considering that individual chapters (2-11) are allocated for specific categories and types of materials in the AACR2R, it is recommended that chapter 4 focus more on cataloguing of manuscript collections while the cataloguing of parts of a collection and individual items be discussed under chapters 2-11 in conjunction with chapter 13 (analysis). Greater distinction (or similarities reconciled) need to be drawn between
individual manuscript cataloguing described under chapter
4 and unpublished materials described in the individual
chapters, particularly, chapter 2 of the code.

- That with respect to manuscripts, the AACR2R include
extensive notes to cover aspects covered in traditional
archive and manuscript practice. When doing so, the
terminology and terms used for manuscripts should form
part of the vocabulary.

- That the AACR2R set aside specific notes in the note
area for local data such as classification numbers,
identification numbers and accession numbers. Optionally,
these details may appear outside the note in
pre-determined positions of a catalogue entry.

- That the AACR2R analytical structures be extended to
reflect more than just two levels of a collection. Use of
a relationship complexity note is recommended to fully
describe a collection or portions of it. In addition,
accession numbers or any means used to identify the part
of a collection being described, should be included in a
note to reflect the relationship of a part to it's whole
within and between institutions/repositories.
- Considering that machine-readability is one of the objectives of the ISBD, it is recommended that a more rigid set of rules be developed to make the ISBD machine-readable so that individual elements can be extracted and identified algorithmically or mechanically. These set of rules need not necessarily match the way the ISBD is output or displayed. A user should have the option to view the display or output with or without the characters identifying the elements in an entry.

- The AACR2R has failed to take cognizance of the way catalogues are searched and sorted, for example, the inclusion of honorifics before or after the initials/forenames upsets the normal sort order and the headings as outlined in the code cannot be used as a sort key. Techniques of hashing names, for instance, 'Lawrence, David Herbert' into 'Lawrence, D H' do not form part of the code. Similarly, searching on the partial form, for example, 'lawren' or 'lawren?, d h' (Wajenberg, 1990:495) to retrieve 'Lawrence, David Herbert' is not documented in the AACR2R, but is in use today. A code should standardise the methods and techniques of searching. The formulation of useful guidelines for searching would facilitate standardisation and ensure that there is a greater coincidence between
the concepts constructed during the cataloguing process and the concepts and techniques used during the search. Hash codes developed in the study for the titles, names, corporate bodies and place come a small way in exploiting the variety of search and access options that are possible.

- It is recommended that a minimum standard be developed for elements to be included in NAREM. In addition, it is recommended that NAREM accept more than one level of description for participating members to adequately serve as a national database for manuscripts. Attention need to be given to important documents in a collection through detailed descriptions as catalogue entries, guides and lists which should be made available online.

- The NAREM and ISBD formats are just two possible output formats. Computer-based catalogues have the potential of being arranged in a variety of formats including the MARC, a labelled format, brief, columnar and other formats to satisfy different user needs. A code such as the AACR2R should go beyond the ISBD format and include, for instance, a labelled format and columnar format that are in frequent use today.
- That studies be undertaken for manuscripts of all types of materials, including, inter alia, cartographic material, music sheets, audio-visual material and computer files. The final objective of such studies would be to arrive at a composite model for manuscripts and unpublished material.

- That a common conceptual framework for all types of materials be developed which would move beyond the limitations of the present study which has limited itself to only manuscripts. An integrated catalogue incorporating both published and unpublished material and different kinds of materials and different kinds of finding aids with the objective of serving different kind of user needs, would bring together the library and archival profession under one roof rather than them existing as separate professions with separate needs.

- The study has focused primarily on description rather than access. By recognising that description is access (Duke, 1989:123), greater use of guides, indexes, lists and calendars for instance, which are the tools used in archival and manuscripts, is envisaged. It is recommended that the E-R model proposed in the study be reviewed to develop suitable mechanisms to include the searching of
finding aids used by archivists. In doing so it would allow for different kinds and levels of access to materials in libraries and archives.

- Modelling techniques have been limited to E-R modelling and data normalisation. These approaches to data analysis require further investigation, to determine their suitability with respect to the diversity of bibliographic elements and relationships in a wider bibliographic environment that includes manuscripts and archives. It is hoped that new approaches to data analysis would lead to new insights and record structures that are in compliance with cataloguing codes and bibliographic principles.

CONCLUSION
The study has demonstrated that in terms of descriptive elements both the library and archival professions can come together in serving a common goal, that of providing records that can be integrated into one database. If however, the principles and practices of both the professions are combined into one body of principles then a unitary approach to cataloguing and describing materials and the provision of a diversity of information
services can be more realisable. Divergent approaches practices and techniques can be drawn from a common set of principles. This is possible in the light of the fact that archives adopt library techniques and libraries adopt archival principles when cataloguing and describing their materials. A common ground can be reached.

Combining all manuscript materials into one database means that a searcher will be able to use just one search strategy (with different options) to identify both item and collection level records in one database while producing reports for different purposes. A unitary approach to manuscript cataloguing can serve to foster greater links and co-operation between the various professions and institutions engaged in preserving historical documents and manuscripts. The study has come a short distance in meeting this objective.

Integration can go one step further. Searching for a book on bridge for instance can lead the user to the text, a short summary on the topic or even make the user aware of the local bridge club (Layne, 1989:188). People do not know what exists, nor are they necessarily aware of what exists until they see it in front of them. For this reason, integrating the variety of functions, materials
and services within and outside the institution can expose the user to a wider variety of choices and a richer experience in searching the catalogue. Ideally a user going to the catalogue should find useful objects they were looking for and related useful objects they had not known of previously (Layne, 1989:189).

Up to now database design has been the the domain of computer science and little attention has been given to database design and database management systems as part of the discipline of information science. The teaching of DBMS and database design in library schools would not only introduce new areas to be incorporated in the discipline, but would hopefully, lead to the design of complex DBMS systems and record structures that are consistent with library and archival practices.

'The focus in designing future catalog records should be on the common bibliographic principles that can unify all catalogs. The code must be built upon the commonality of catalog structure, not necessarily upon any specific formal implementation' (Duke, 1989:119).
Similarly, the inclusion of record management principles and practices such as selection, arrangement, description, preservation, sorting and compilation of various lists and registers in library schools curricula can serve to bridge the gap between the professions.

However, well intended a code, phrases such as 'if appropriate', 'if important', 'commonly known', 'predominant or distinctive', leave the code subject to decisions, and if major cataloguing agencies and bodies involved in code development cannot arrive at a decision, how can the cataloguer in the local context be expected to apply the code consistently? (Ayres, 1980:776). The differences between interpretations will always be there. Computerisation can bridge the gap by making more than one relationship possible, and at the same time limit the alternatives which can form the framework for local, regional, national and international co-operation. This is done against the background that the task of codification is to create flexible standards for individual local interpretations, while doing away with the necessity for petty local variations (Ayres, 1980:776).
The basic function of the catalogue would not change, but the catalogue of the future coupled with automation would become more sophisticated ranging from finding of relevant sources, to document retrieval, authority control and provision of community information for both published and unpublished works.

The future must embrace the coexistence of authorship, main entry and provenance, bring together the different forms of material, develop a common set of standards for description (not necessarily identical but compatible), enhance the quality of catalogues and bring both seemingly divergent professions together.

In the final analysis, the combination of sophisticated underlying data structures of DBMS, a more relevant code, greater participation by librarians, archivists and information scientists in specifying their requirements and greater use of computer technology would pave the way to an integrated catalogue of the future.
APPENDIX A

LIST 1
- braille
- cartographic material
- computer file
- graphic
- manuscript
- microform
- motion picture
- multimedia
- music
- object
- sound recording
- text
- videorecording

LIST 2
- art original
- art reproduction
- braille
- chart
- computer file
- diorama
- filmstrip
- flash card
- game
- globe
- kit
- manuscript
- map
- microform
- microscope slide
- model
- motion picture
- music
- picture
- realia
- slide
- sound recording
- technical drawing
- text
- toy
- transparency
- videorecording

GENERAL MATERIAL DESIGNATION (GMD) CODES (AACR2R, 1988:21)

395
APPENDIX B - EXAMPLE OF AN INVENTORY

Accession No: KCM 89/18

Produced by the
Ellie Campbell Africana Library
for research purposes

The Mildred Lavoipierre Collection consists of 143 files containing letters, personal and official, and papers relating mainly to the various organisations to which she belonged. In the collection are minutes of meetings, notices, agendas of meetings, constitutions, etc of organisations such as the Durban Bantu Child Welfare Society, the S A National Council for Child Welfare, the National Council of Women of South Africa and the Council for Human Rights.

There are also files under various headings such as malnutrition, the Effingham Rail Disaster Relief Fund, begging and housing.

The contents of this collection make a valuable contribution to the social history of Durban.

Biographical notes:

Mildred Burns Lavoipierre (née Hughes) was born in Pietermaritzburg on 16 July 1907. She moved to Durban during the 1914/18 war and was educated at the Durban Girls' College and then the Convent High School, St Andrews Street, Durban. Known to her friends as Pat, Mrs Lavoipierre qualified as a commercial teacher, and after 10 years teaching at the Natal Technical College, opened her own school, the Commercial Training College, in 1937.

In 1941, she married Jean Lavoipierre, who was 16 years her senior. They had no children and he died in 1953.

In 1944, Mrs Lavoipierre was co-founder of the first Place of Safety and Detention for Bantu, Indian and Coloured children in Durban. From the early forties she was a member of the Durban Bantu Child Welfare Society, the Durban Child Welfare Society and the S A National Council for Child Welfare.

In 1958 she was accorded Civic Honours by the City of Durban and in 1966 was presented with the "Woman of the Year" award by the Durban Branch of the Union of Jewish Women of South Africa.

Mildred Lavoipierre died in April 1984. Although she was a teacher by profession, not a social worker, she made an outstanding contribution to child welfare in particular to Bantu child welfare, in Durban.

For further information see File 1.

NCAL
September 1990
50/18
<table>
<thead>
<tr>
<th>File no.</th>
<th>Item no.</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCM 89/18/1</td>
<td>KCM 89/13/1/1-7</td>
<td>Biographical description, testimonial, interviews &amp; newscuttings. MSS, TS. News cuttings. PY of News cuttings.</td>
<td>8 Apr 1937 - 1984</td>
</tr>
<tr>
<td>2</td>
<td>89/14/2/1</td>
<td>Diary 1942. MS</td>
<td>Oct 1942 - 23 Feb 1943</td>
</tr>
<tr>
<td>3</td>
<td>89/18/3/1-26</td>
<td>Speeches, lectures, reports given by Mildred Lavoipierre. MSS, TSS, CC of TSS, News cuttings.</td>
<td>[19467]-1972</td>
</tr>
<tr>
<td>4</td>
<td>89/18/4/1-23</td>
<td>Speeches made by Mildred Lavoipierre. MSS, TSS, CC of TSS.</td>
<td>Undated</td>
</tr>
<tr>
<td>5</td>
<td>89/18/5/1-8</td>
<td>Tributes, references written by Mildred Lavoipierre. MSS, TSS, CC of TSS, printed material.</td>
<td>1961-1977 &amp; undated</td>
</tr>
<tr>
<td>6</td>
<td>89/18/6/1-4</td>
<td>Correspondence, invitation &amp; agenda from the Durban City Council for Civic Award. MSS, printed material.</td>
<td>19 Jun 1958 - 21 Nov 1958</td>
</tr>
<tr>
<td>89/18/6/5-22</td>
<td>Letters to M.B.L. re Civil Honours. MSS, TSS, printed material.</td>
<td>17 Jun 1958 - 16 Jan 1959</td>
<td></td>
</tr>
<tr>
<td>89/18/6/22</td>
<td>Letter to M.A.L. from Union of Jewish Women of Southern Africa re &quot;Woman of the Year&quot; Award. TS.</td>
<td>25 Jul 1960</td>
<td></td>
</tr>
<tr>
<td>89/18/6/24-33</td>
<td>Correspondence, etc. concerning Union of Jewish Women &quot;Woman of the Year&quot; Award to M.B.L. MSS, TSS, printed material.</td>
<td>13 Oct 1960 - [1977]</td>
<td></td>
</tr>
<tr>
<td>89/18/6/35-37</td>
<td>Notes of congratulations to Mildred Lavoipierre. MSS</td>
<td>Undated</td>
<td></td>
</tr>
</tbody>
</table>

397
<table>
<thead>
<tr>
<th>No.</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>27 Mar 1948</td>
<td>Letter from M.B.L. to &quot;the Editor, The Daily News&quot;, Durban re Bantu child living in Woodlands, CC of TS.</td>
</tr>
<tr>
<td>9</td>
<td>4 May 1953</td>
<td>Letters of condolence sent to M.B. Lavolpierre on the death of her husband, Jean Lavolpierre. MSS, TSS.</td>
</tr>
<tr>
<td>10</td>
<td>17 May 1940</td>
<td>Letter, notes to M.B. Hughes (Lavolpierre) from the Union Unity Fund. MSS, TSS.</td>
</tr>
<tr>
<td>87</td>
<td>20 Oct 1942</td>
<td>Letters (personal &amp; semi-official) sent to Mildred Lavolpierre. MSS, TSS.</td>
</tr>
<tr>
<td>88</td>
<td>3 Mar 1947</td>
<td>Letter from Joshua Nkomo, Pietermaritzburg, to Mildred Lavolpierre, Durban. MS.</td>
</tr>
<tr>
<td>89</td>
<td>13 Sep 1947</td>
<td>Letters (personal &amp; semi-official) sent to Mildred Lavolpierre. MSS, TSS.</td>
</tr>
<tr>
<td>90</td>
<td>18 Jan 1958</td>
<td>Letter from Jordan K Nyubane, Shanda, Natal to Mrs Lavolpierre, (Durban?), TS.</td>
</tr>
</tbody>
</table>
APPENDIX C - MAIN MODULE PROGRAMS

ORDER OF PROGRAMS:

1. APROG - Topmost Program
2. IPROG - Input Program
3. UPROG - Update Program
4. DPROG - Deletion Program
5. AREP - AACR2R Report Program
6. NREP - NAREM Report Program
7. REINDEX - Reindexing Program
8. DEPAUT - Depot Authority Program

(The Packing Utility is found in Aprog)
** Program Name : Aprog
** Date : 31/12/93
** Written by : K. Chetty
** Aim : Topmost menu with options to create, update, delete catalogue records. The menu also has options to generate AACR2R and NAREM reports, a re-index utility to reindex files, a utility to pack files and a depot authority to create, update and delete depot records.

Set Talk Off
Set Bell Off
Set Status Off

Do While .t.
Close Databases
Set Procedure to Proc1
Clipper = .T.
Asel = '0'

Clear
Skelvar = 'T O P M O S T C A T A L O G U I N G M E N U'
Do Skeleton with Skelvar, 'Y', 'APROG SCREEN'
@6,20 Say '1. Cataloguing Input'
@7,20 Say '2. Cataloguing Update'
@8,20 Say '3. Record Deletion'
@9,20 Say '4. Generate AACR2R Report'
@10,20 Say '5. Generate NAREM Report'
@11,20 Say '6. Reindex Files'
@12,20 Say '7. Pack Files'
@13,20 Say '8. Depot Authority File Maintenance'
@14,20 Say '0. Exit Program'
@16,20 Say 'Select : ' Get Asel
Read

Do Case
Case Asel = '0'
  Exit

Case Asel = '1'
  Do Iprog
     & & Enters new records
Case Asel = '2'
  Do Uprog
     & & Updates records
Case Asel = '3'
  Do Dprog
     & & Deletes and recalls records
Case Asel = '4'
  Do Arep
     & & Generates AACR2R Report
Case Asel = '5'
  Do Nrep
     & & Generates NAREM Report

400
Case Asel = '6'
      Do Reindex        && Reindexes Table1 and Depot files
Case Asel = '8'
      Do Depaut        && Depot Authority File Maintenance
Case Asel = '7'
      && Packs Table1 and Depot files

Use Table1
Skelvar = 'PACKING UTILITY'

DO skeleton with Skelvar, 'Y', 'PACK SCREEN'
@ 7,0 Say 'Packing Table1'
@ 7,45 Say 'Number of records = ' + Str(Reccount())
Pack
@ 8,0 Say 'Packing complete'
@ 8,45 Say 'Number of records = ' + Str(Reccount())
@10,0 Say 'Please reindex Table1 using Reindex option of Topmenu'

** Packing Depot File
Use Depot Index Depot
@13, 0 Say 'Packing Depot File'
@13, 45 Say 'Number of records = ' + Str(Reccount())
Pack
@14,0 Say 'Packing and Reindexing Complete'
@14,45 Say 'Number of Records = ' + Str(Reccount())
Wait
Close all
Set Procedure to Procfl
   Endcase
Enddo
Return
**Program: Iprog
** Written by K Chetty
** Date 12/30/93

** Setting Environmental Details
Close Databases
Close All
Clear
Set Procedure to Procfl
Select 1

Use Table1 index Table1, Ref, Tkey, Mkey, Ukey, Maindex
Select 2
Use Depot index Depot
Select 1

Do while .T.
 Progname = 'IPROG'
 Brnx = 0
 Mainx = Space(1)
 Mkeyx = Space(6)
 Mtypex = Space(1)
 Uniformx = Space(1)
 Utypex = Space(1)
 Ukeyx = Space(6)
 Msortx = Space(1)
 T_artx = Space(10)
 Titledx = Space(254)
 Tkeyx = Space(6)
 Edx = Space(50)
 Datx = Space(30)
 Phdescx = Space(60)
 Classx = Space(20)
 Memx = Space(254)
 Connotex = Chr(181)
 Commemx = Space(1)
 Refx = Space(20)
 Depotx = Space(20)
 Naremx = .N.
 Typep = 'M'
 Date1x = '00000000'
 Date2x = '00000000'

**** Entering Title details
 Keyx = Space(6)
 Do Ititle && Procedure inputs title and filing article
   If Len(Trim(Titledx)) =0
     Exit
   Endif
 Titledy = Titledx
 Do wcomp2 with Titledy, Keyx && Generates Tkeyx
Tkeyx = keyx

** Validates Title and outputs Duplicate records
Select 1
Set order to 3 && Tkey made the primary search key
Go Top
Interrupt = 'C'
Seek Keyx
Do While .Not. Eof() .and. Tkey = Keyx
If keyx = Space(6)
Exit
Endif

Isbdx = Space(1)
Do ISBD0B && Brief form of ISBD is output (excludes notes)
Clear
Skelvar = 'DUPLICATE TITLES'
Do Skeleton with Skelvar, 'Y', 'DUPLICATE TITLES'
@8,0 Say Isbdx
@22,0 Say 'Continue search, Skip Searching, Abandon new Record Creation : (C/S/A)';
Get Interrupt Picture '!'
Read
If Interrupt = 'A'
Exit
Endif
If Interrupt = 'S'
Exit
Endif
Skip
Enddo

Select 1
Set order to 1
If Interrupt = 'A'
&& Exits input menu
Exit
Endif

Do Main && Enters Main Entry Details

Do Screen3 && Enters edition, dates, physical details
&& Reference No., and Depot Code and validates accordingly

Do Notes && Notes entered

Do Conmem && Concatenates all fields into variable Conmemx which is
Select 1 && Saved as field Mem in Table1
Set Order to 1
Go Bottom && Find highest Brn
Brnx = Brn + 1 && Assign next highest Brn to current record
Append Blank
Replace Brn with Brnx

403
Clear
Skelvar = 'INPUT MENU'
@14,10 Say 'ASSIGNING NEW BRN'
@16,10 Say 'Your New BRN is: ' Get Brnx
? 'Replacing fields in Table1'
Replace Brn with Brnx
Replace Ref with Refx
Replace Depot with Depotx
Replace Type with Typex
Replace T_art with T_artx
Replace Mkey with Mkeyx
Replace Mtype with Mtypepx
Replace Msrt with Msrtx
Replace Tkey with Tkeyx
Replace Utype with Utypepx
Replace Ukey with Ukeyx
Replace Date1 with Date1x
Replace Date2 with Date2x
Replace Narem with Naremx
Replace Class with Classx
If Clipper
Replace Mem with Conmemx
Endif
wait
Enddo
RETURN
**Program Uprog.prg
** Update Program
** Written by K Chetty
** Date 12/31/93

Clear
Close All
Set Procedure to Procfl
Progname = 'UPROG'

Select 1
Use Table1 Index Table1, Ref, Tkey, Mkey, Ukey, Maindex
Select 2
Use Depot Index Depot
Select 1

Topflag = 'T' && Inner do while loop (below) continues until
&& valid Brn entered

Do While .T.
  Do While .T.
    Topflag = 'T'
    Brnx = 0
    Skelvar = 'UPDATE PROGRAM'
    Do Skeleton with Skelvar, 'Y', 'UPDATE SCREEN'

    @12,12 Say "ENTER BRN, TO EXIT SET BRN TO 0; " Get Brnx
    Read
    If Brnx = 0
      Exit
    Endif
    Go Top
    Seek Brnx
    If Found()
      Isbdx = Space(1)
      Do Isbdxb && Displays brief details on screen
      Do Skeleton with Skelvar, 'Y', 'UPDATE VIEW SCREEN'
      @5,0 Say Isbdx
      Wait

    ** Variables filled with contents of record in Table1
      Refx = Ref
      Brnx = Brn
      Depotx = Depot
      Typepx = Type
      Mkeyx = Mkey
      Mtypex = Mtype
      Msrttx = Msort
      T_artrx = T_art
      Tkeyx = Tkey
      Utypex = Utype

505
Ukeyx = Ukey
Date1x = date1
Date2x = date2
Narremx = Narem
Classx = Class
E1 = Space(1)
E2 = Space(1)
E3 = Space(1)
E4 = Space(1)
E5 = Space(1)
E6 = Space(1)
E7 = Space(1)
Connotex = Space(1)
Conmemx = Space(1)

Do Ucommem  & & Fills all variables E1-E7 from field Mem
  Mainx = E1
  Uniformx = E2
  Titlex = E3
  Edx = E4
  Datx = E6
  Phdescx = E5
  Connotex = E7  & & Notes kept in storage format
  If Len(Trim(Connotex)) < 2
    Connotex = Space(1)
  Endif
Else
  @14,12 Say 'Brn not Found, Retry'
  Topflag = 'N'
  WAIT
Endif
Exit
Enddo

Dopt = 'A'
Do while .T.
  Clear
  If Topflag = 'N'
    Exit  & & Exits if record not found
  Endif
  If Brnx = 0
    Exit  & & Skips updating
  Endif

** Setting up Screen
@5,0 Say ""
Text
  A. Title and Statement of Responsibility
  B. Main Entry Headings

406
C. Other Fields
D. Notes
E. Exit Update

Endtext
Do Skeleton with Skelvar, 'N', 'UPDATE OPTION SCREEN'
@18,0 Say 'Select Option : ' Get Dopt Picture '!
Read
Clear

Do Case
  Case Dopt = 'A'    && Updates Title
    Do Ititle
      Tkeyx = Space(6)
      Keyx = Space(6)
      Titley = 'Ttitle'
    Do Wcomp2 with Titley, Keyx    && Key constructed
    Tkeyx = Keyx
  Case Dopt = 'B'    && Main Entry Heading Updated
    Do Main
  Case Dopt = 'C'    && Data such as Depot code, Reference
    Do Screen3
      && Class no and physical description updated
  Case Dopt = 'D'
    Do Notes    && Notes updated
  Case Dopt = 'E'
    ** On exit fields reconstructed for storage
    Do Commem
      && Mem field constructed
    Select 1
    Set order to 1
    Seek Brnx
    If Found()
      Replace Ref with Refx
      Replace Depot with Depotx
      Replace Type with Typex
      Replace T_art with T_artx
      Replace Mkey with Mkeyx
      Replace Mtype with Mtypex
      replace Tkey with Tkeyx
      Replace Utype with Utypex
      Replace Ukey with Ukeyx
      Replace Date1 with Date1x
      Replace Date2 with Date2x
      Replace Narem with Naremx
      Replace Class with Classx
  Endcase

407
Replace Msort with Msortx
If Clipper
   Replace Mem with Conmemx
   Endif
Else
   Clear
   ?'Fatal Error, Updated Record not found'
   wait
   Endif
Exit
Endcase
Enddo

If Brnx = 0
   Exit && Exits outerloop and Returns to Aprog
   Endif
Enddo
RETURN
** Program Name : Dprog.prg
** Date 31/12/93
** Written by K. Chetty
** Aim : Deletes and recalls/reinstates deleted records of Table1

Select !
Use Table1 Index Table1, Ref, Tkey, Mkey, Ukey
Select 2
Use Depot index Depot
Select 1
Do while .t.
Clear
Skelvar = 'DELETE OPTION'
Do Skeleton with Skelvar, 'Y', 'DELETION SCREEN'
Brnx = 0
@5,5 Say 'Enter BRN (0 to exit) : ' Get Brnx
Read
If Brnx = 0
Exit
Endif
Dopt = 'Y'
@7,5 Say 'BRN selected is ' + Str(Brnx)
@8,5 Say 'Confirm deletion (Y, to confirm, N to reject) : ' Get Dopt Picture '!'
Read
If Dopt = 'Y'
@10,5 Say 'Confirmation accepted - checking if record exists'
Go Top
Seek Brnx
If Found()
  If Deleted()
    Dopt2 = 'Y'
    @11,5 Say 'Item already deleted'
    @12,5 Say 'Undelete/Recall Item (Y/N) : ' Get Dopt2 Picture '!
    Read
    If Dopt2 = 'Y'
      Recall
      @13,5 Say 'Item UNDELETED/RECALLED'
    Endif
    Else
      Delete
      @11,5 Say 'Deletion successful'
    Endif
  Else
    @12,5 Say 'Item not found, not deleted'
  Endif
Else
  @11,5 Say 'Failed to receive confirmation'
  @12,5 Say 'Record not deleted'
Endif
wait
Enddo
Clear
Do Skeleton with Skelvar, 'Y', 'DELETION SCREEN'
@6,0 Say '"
? 'USE THE PACK UTILITY TO REMOVE DELETED RECORDS.'
? 'WHEN PACKING IS COMPLETE, REINDEX THE RECORDS'
? 'USING THE REINDEX UTILITY, OPTION 6, OF APROC/TOPMENU.'
Wait
Close Databases
RETURN
**Program: Arep**
**Written by K Chetty**
**12/31/93**
**Aim: To print ISBD reports**

Do While .T.
Select 1
Use Table1 Index Table1
Select 2
Use Depot Index Depot
Set Procedure to Procfl

Skelvar = 'ISBD REPORTS'
Heading = Space(1)
Private Anyflag
Anyflag = 'T'
Qopt = 'C'
Mainindex = '
Do Skeleton with Skelvar, 'Y', 'ISBD SCREEN'
Clipper = .T.

@12,4 Say 'Please ensure that the headings have been sorted'
@14,4 Say 'Sort Heading using option 3 of Reindexing Menu'
@16,4 Say 'Continue/Quit Reporting (C/Q): ' Get Qopt Picture '!
Read
If Qopt = 'Q'
Return
Endif

Topflag = .T.

Do While .T.
If Topflag
Else
Wait && Halts screen to view messages
Endif
Topflag = .T.
Popt = 'N'
Clear
Skelvar3 = 'ISBD SCREEN'
Do Skeleton with Skelvar, 'Y', Skelvar3

Depotx = Space(20)
Brnx1 = 1
Select 1
Go Bottom && Finds highest Brn
If .Not. Eof()
Mbrnx = Brn
Else
Mbrnx = 1
Endif

411
Brnx2 = Mbrnx

@8, 10 Say 'Enter Source Code : ' Get Depotx
@10,10 Say 'Print (Y/N) : ' Get Popt Picture '!'
@12,10 Say 'Brn Range : ' Get Brnx1
@12,50 Get Brnx2
@16,10 Say 'To exit leave Depot code Blank. Set Brn to 0'
@17,10 Say 'Use Brn range to print selected reports'
Read

If Len(Trim(Depotx)) = 0 .And. Brnx1 = 0
   Return
Endif

If Brnx1 > Brnx2
   @23,0 Say Space(80)
   @23,0 Say 'High Brn less than lower Brn - High Brn reset to lower Brn'
   Brnx2 = Brnx1
Endif

If Mbrnx < Brnx2
   @23,0 Say Space(80)
   @23,0 Say 'High Brn greater than Brn in system - High Brn reset to Max Brn'
   Brnx2 = Mbrnx
Endif

If Len(Trim(Depotx)) = 0
   @21,0 Say 'Depot Code has been omitted. code not used for selection'
   Topflag = .N.
Endif

Dcodex = Upper(Depotx)
Depflag = .T.
Go Top

If Len(Trim(Depotx)) = 0
   Exit
Else
   Do Depotval && Validates Depot code
   If Depflag
      Exit
   Else
      @21,0 Say 'Invalid Depot Code entered'
      Topflag = .N.
   Endif
   Enddo

Enddo

** Displays or Prints Records using maindex as controlling index
** Since entries are required sorted by heading and no selection
** Criteria is stipulated (except Brn and Depot) the index **
** Loops from the beginning to the end and tests for the depot and Brn **
** within the loop ensuring that the printed report is sorted by heading **

If Popt = 'Y' .or. Popt = 'y'  
Set Print On  
Endif

** Displaying Headings one by one in sorted order **
Use Table1 index mainex  
Set order to 1  
Go Top  

Go Top  
Do while .not. EOF()  

If Upper(Depotx) = Upper(Depot) .or. Len(Trim(Depotx)) = 0  
If Brn >= Brnx1 .and. Brn <= Brnx2  
    Brnx = Brn  
    Mainx = Space(1)  
    Mainy = Space(1)  
    Mkeyx = Mkey  
    Mtypep = Mtype  
    Utypep = Utype  
    Ukeyx = Ukey  
    Tkeyx = Tkey  
    If Clipper  
    Memx = Mem  
    Endif

    Connotex = Space(1)  
    Conmemx = Space(1)  
    E1 = Space(1)  
    E2 = Space(1)  
    E3 = Space(1)  
    E4 = Space(1)  
    E5 = Space(1)  
    E6 = Space(1)  
    E7 = Space(1)  
    Do Ucommem && Fills E1-E7 with contents of Mem field of Table1  
    Connotex = E7  

** ISOLATING UNIFORM TITLES **
Uel1 = Space(1)  
Uel2 = Space(1)  
Uniformmx = E2  
Element1lx = Space(1)  
Addsx = Space(1)  
Upunct = ','

If Len(Trim(Uniformmx)) > 3 && String contains at least 3 delimiters  
Upunct = Substr(Uniformmx,Len(Uniformmx)-1,1)
If Upunct $'.,('
Else
Upunct = '('
Endif
Uniformx = Substr(Uniformx, 2, Len(Uniformx) - 1)
Element11x = Substr(Uniformx, 1, At(chr(181), Uniformx) - 1)
Element11x = Element11x + Space(100 - Len(Element11x))
Addsl1x = Substr(Uniformx, At(chr(181), Uniformx) + 1, Len(Uniformx) -
At(chr(181), Uniformx) - 3)
Addsl1x = Addsl1x + Space(50 - Len(Addsl1x))
Else
Element11x = Space(100)
Addsl1x = Space(50)
Upunct = '('
Endif

Uel1 = Element11x
Uel2 = Addsl1x

** Stripping Main Heading Elements
E11 = Space(100)
E12 = Space(50)
E13 = Space(50)
E14 = Space(1)
E15 = Space(1)
E16 = Space(1)
If Len(Trim(E1)) > 1
Mainy = E1
Do Uconmain
Endif

If Len(E16) = 0
E16 = ' '
Endif

Titlex = Ltrim(Trim(T_art) + ' ' + Trim(E3))
***Titlex = Trim(Titlex) + Space(254 - Len(Trim(Titlex)))
Lnel = Space(80)
Lne2 = Space(80)

Do Case
Case Utype = 'U' && Uniform Titles exist
If Mtype = 'A' && Author exist
Do Case
Case E16 = 'Y' && If British true
  Anyflag = 'T'
  If Len(Trim(E14)) > 0 && Nobility exists
    Lne1 = Trim(E11) + ' ' + '(' + Trim(E14) + ')' +
  Anyflag = 'N'
Else
  Lne1 = Trim(E11)
Endif

414
Anyflag2 = 'Y'
If Len(Trim(E12)) > 0 && Forenames exist
   If Anyflag = 'N'
      Lnel = Ln1 + ' ' + Trim(E12)
   Else
      Ln1 = Ln1 + ',', '+ Trim(E12)
   Endif
   Anyflag2 = 'N'
Endif

If Len(Trim(E13)) > 0 && Additions exist
   If anyflag2 = 'N'.and. Anyflag = 'N'
      Ln1 = Ln1 + ',', '+ Trim(E13)
   Else
      If Anyflag = 'N'
         Ln1 = Ln1 + ',', '+ Trim(E13)
      Else
         Ln1 = Ln1 + ',', '+ Trim(E13)
      Endif
   Endif
Endif

If Len(Trim(E15)) > 0 && Relator codes exist
   Ln1 = Trim(Ln1) + ',', '+ Trim(E15)
Endif

Lne2 = Trim(Ue11)
If Len(Trim(Ue12)) > 0 && Uniform additions exist
   If Upunct = '
      Lne2 = Lne2 + ' ( '+ Trim(Ue12) + ')' 
   Else
      Lne2 = Lne2 + Upunct + ',', '+ Trim(Ue12)
   Endif
Endif

Case E16 = 'N'.or. E16 = ' ' && British blank/not true
   Ln1 = Trim(E11)
   If Len(Trim(E12)) > 0 && Forenames exist
      Ln1 = Ln1 + ',', '+ Trim(E12)
   Endif

   If Len(Trim(E14)) > 0 && Nobility exists
      Ln1 = Ln1 + ',', '+ Trim(E14) + ')
   Endif

   If Len(Trim(E13)) > 0 && Additions exist
      Ln1 = Trim(Ln1) + ',', '+ Trim(E13)
   Endif

   If Len(Trim(E15)) > 0 && Relator codes exist
      Ln1 = Trim(Ln1) + ',', '+ Trim(E15)

415
Endif
Lne2 = Trim(Uel1)
If Len(Trim(Ue2)) > 0 && Uniform additions exist
   If Upunct = '('
      Lne2 = Lne2 + '(' + Trim(Ue12) + ')
   Else
      Lne2 = Lne2 + Upunct + ' ' + Trim(Ue12)
   Endif
Endif
Endcase
Endif

If Mtype = 'P' && Place exists
   If Len(Trim(E13)) > 0 && Additions exist
      Lnel = Trim(E11) + '(' + Trim(E13) + ')
   Else
      Lnel = Trim(E11)
   Endif

Lne2 = Trim(Uel1)
If Len(Trim(Ue12)) > 0 && Uniform additions exist
   If Upunct = '('
      Lne2 = Lne2 + '(' + Trim(Ue12) + ')
   Else
      Lne2 = Lne2 + Upunct + ' ' + Trim(Ue12)
   Endif
Endif
Endif

If Mtype = 'C' && Corporate Body exists
Lnel = Trim(E11)
If Len(Trim(E13)) > 0 && Additions exist
   Lnel = Lnel + '(' + Trim(E13) + ')
Endif

Lne2 = Trim(Uel1)
If Len(Trim(Ue12)) > 0 && Uniform additions exist
   If Upunct = '('
      Lne2 = Lne2 + '(' + Trim(Ue12) + ')
   Else
      Lne2 = Lne2 + Upunct + ' ' + Trim(Ue12)
   Endif
Endif
Endif

If Mtype = '/' && Only uniform titles exist
Lnel = Trim(Uel1)
If Len(Trim(Ue12)) > 0 && Uniform additions exist
   If Upunct = '('
      Lnel = Lnel + '(' + Trim(Ue12) + ')
Endif

416
Else
    Lnl1 = Lnl1 + Upunct +'' + Trim(Uel2)
Endif
Endif
Lne2 = ''
Endif

Case Utype = 'T' && Title is main entry
    Lnel = Trim(Titlex)
    If len(Lnel) > 1
        If '/ ' $ Lnel
            Lnel = Substr(Lnel,1,At('/ ',Lnel)-1)
        Endif
    Endif
    Lne2 = ''
Endif

Case Utype = '' && Uniform title does not exist
    If Mtype = 'A' && Author exists
        Do Case
            Case E16 = 'Y', && British is true
                Anyflag = 'T'
                If Len(Trim(E14)) > 0 && Nobility exists
                    Lnel = Trim(E11) + '' + '(' + Trim(e14) + ')
                    Anyflag = 'N'
                Else
                    Lnel = Trim(E11)
                Endif
            Endif

            Anyflag2 = 'Y'
            If Len(Trim(E12)) > 0 && Forenames exist
                If Anyflag = 'N'
                    Lnel = Lnel + '' + Trim(E12)
                Else
                    Lnel = Lnel + '' + Trim(E12)
                Endif
                Anyflag2 = 'N'
            Endif

            If Len(Trim(E13)) > 0 && Additions exist
                If anyflag2 = 'N', && Anyflag = 'N'
                    Lnel = Lnel + '' + Trim(E13)
                Else
                    If Anyflag = 'N'
                        Lnel = Lnel + '' + Trim(E13)
                    Else
                        Lnel = Lnel + '' + Trim(E13)
                    Endif
                Endif
            Endif
        End Case
    Endif
Endif

If Len(Trim(E15)) > 0
    Lnel = Trim(Lnel) + '', '' + Trim(e15)

417
Endif

Lne2 = Space(1)  && No second level heading

Case E16 = 'M'.or. E16 = ''  && British blank/not true
  Lnel = Trim(E11)
  If Len(Trim(E12)) > 0  && Forenames exist
    Lnel = Lnel + ', ' + Trim(E12)
  Endif

  If Len(Trim(E14))>0  && Nobility exists
    Lnel = Lnel + ', (' + Trim(E14) + ')
  Endif

  If Len(Trim(E13)) > 0  && Additions exist
    Lnel = trim(Lnel) + ', ' + Trim(E13)
  Endif

  If Len(Trim(E15)) >0  && Relator codes exist
    Lnel = Trim(Lnel) + ', ' + Trim(E15)
  Endif

Lne2 = Space(1)  && No second level heading
Endcase
Endif

If Mtype = 'P'  && Place exists
  If Len(Trim(E13)) > 0  && Additions exist
    Lnel = Trim(E11) + ' (' + Trim(E13) + ')
  Else
    Lnel = Trim(E11)
  Endif
Lne2 = Space(1)  && No second level heading exists
Endif

If Mtype = 'C'  && Corporate body exists
  Lnel = Trim(E11)
  If Len(Trim(E13))>0  && Additions exists
    Lnel = Lnel + ' (' + Trim(E13) + ')
  Endif
Lne2 = Space(1)  && No second level heading
Endif

** No Main Entry Headings
If Mtype = '',
  Lnel = ''  && All lines are set to null
Lne2 = ''
Endif
Endcase

** The next step after constructing the heading (above) is to format
** the entry. Lne01 and Lne02 precedes the AACR2R entry. Lnel and Lne2 are
** First and second lines of the heading while the third formatted element is
** the Isbd (excluding the notes). The notes are displayed individually.
If Popt = 'Y' or Popt = 'y'
Else
Do Skeleton with Skelvar, 'Y', Skelvar3
@20,0 Say Space(80)
@4,0 Say ""
Endif

** Lne01 and Lne02 being formatted
Lne01 = 'Brn : ' + Str(Brn) + Space(10 - Len(Str(Brn))) + Space(19) + 'Source : '
 + Depot
Lne02 = 'Reference : ' + Trim(Substr(Ref, 1, 10)) + Ltrim(substr(Ref, 11, 10)) +
Space(23 - Len(Trim(Substr(Ref, 1, 10)) + Ltrim(Substr(Ref, 11, 10)))) +
'Classification : ' + Class
? Lne01
? Lne02

** First level heading being formatted
Clnel = 0
y = 0
Lnel = Trim(Lnel)
If Len(Trim(Lnel)) > 0
If Substr(Lnel, Len(Lnel), 1) = '.'
Else
Lnel = Lnel + '.'
Endif
Endif

z = Lnel
Lcnt = 1
Do while Lcnt < 5
Lcntx = Ltrim(Str(Lcnt))
W&Lcntx = Space(0)
Lcnt = Lcnt + 1
Enddo

Do Clines with '1', y, z  & first level heading formatted
c1n1el = y

lcnt = 1
DO while lcnt < 5
lcntx = Ltrim(str(lcnt))
If Len(Trim(W&lcntx)) > 0
? W&lcntx
Endif
lcnt = lcnt + 1
Enddo

** Second level heading being formatted
419
c1ne2 = 0
y=0
Lne2 = Trim(lne2)

if Len(trim(lne2)) > 0
    if Substr(lne2,len(lne2), 1) = '.'
        Else
            Lne2 = Lne2 + '.'
        Endif
    Endif

z = lne2
lcnt = 1
Do while lcnt < 5
    lcntx = Ltrim(Str(lcnt))
    W&lcntx = Space(0)
    lcnt = lcnt +1
Enddo

Do Clines with '1', y,z & second level heading formatted c1ne2=y

lcnt = 1
Do while lcnt < 5
    lcntx = Ltrim(str(lcnt))
    If Len(Trim(W&lcntx)) > 0
        n1 = Space(2) + W&lcntx
    Endif
    lcnt = lcnt+1
Enddo

** Constructing ISBD (excluding notes)
Isbd = Ltrim(Trim(Titlex)) & Title added
If Len(Trim(E4))>0
    Isbd = Isbd + ' - ' + Trim(E4) & edition added
Endif
If Len(Trim(E6)) > 0
    ISBD = Isbd + ' - ' + Trim(E6) & date added
Endif
If Len(Trim(E5)) >0
    Isbd = Isbd + ' - ' + Trim(E5) & Physical details added
Endif

If Substr(Isbd,Len(Isbd),1) = '.'
    Else
        Isbd = Isbd + '.'
    Endif

** Formatting ISBD (excluding notes)
c1ne3 = 0
y=0

420
z = Trim(Isbd)
Lcnt = 1
Do while Lcnt < 20
  Lcntx = Ltrim(Str(Lcnt))
  W&Lcntx = Space(0)
  Lcnt = Lcnt + 1
Enddo

Do Clines with '1', y, z    && ISBD formatted - excluding notes
  clne3 = y
  Lcnt = 1
  Do while Lcnt < 20
    Lcntx = Ltrim(str(Lcnt))
    If Len(Trim(W&Lcntx)) > 0
      If Lcnt = 1
        ? W&Lcntx
      Else
        ? Space(4) + W&Lcntx
      Endif
    Endif
    Lcnt = Lcnt + 1
  Enddo

*** Obtaining Notes

P2 = 65
Do While P2 > 64 .and. P2 < 89
  Ax = Ltrim(Str(P2))
  Sx&Ax = Space(1)
  P2 = P2 + 1
Enddo

Do Uconnote

P2 = 65
Do While P2 > 64 .and. P2 < 89
  Ax = Ltrim(Str(P2))
  Sx&Ax = Trim(Sx&Ax)
  If Ax = '81' .or. Ax = '88'
    Else
      If Len(Sx&Ax) < 73 .and. Len(Sx&Ax) > 0
        ? Space(4) + Ltrim(Sx&Ax)
      Else
        Lcnt = 1
        y = 0
        z = Sx&Ax
        Do while Lcnt < 10
          Lcntx = Ltrim(Str(Lcnt))
          W&Lcntx = Space(0)

421
Lcnt = Lcnt + 1
Enddo

Do Clines with '1', y, z  && Each note is formatted
  Lcnt = 1
  Do while Lcnt < 10
    Lcntx = Ltrim(str(Lcnt))
    If Len(Trim(W&Lcntx)) > 0
      If Lcnt = 1
        ? Space(4) + Rtrim(Ltrim(W&Lcntx))
      Else
        ? Rtrim(Ltrim(W&Lcntx))
      Endif
    Endif
    Lcnt = Lcnt+1
  Enddo
Endif

P2 = P2 + 1
Enddo

If Popt = 'Y' .or. Popt = 'y'
Else
  Wait
Endif

P2 = 81
Ax = Ltrim(Str(P2))

If Ax = '81' .and. Len(Trim(Sx&Ax)) > 1
  If Popt = 'Y' .or. Popt = 'y'
    ** Statements to print line by line
    Mline = 1
    noline = M1count(Sx&Ax,65)
    ?
    ? Space(4) + 'SUMMARY :'
    ? Space(4) + '--------'

  yyy = Sx&ax + Chr(13) + Chr(10)
  Do while Len(yyy) > 0
    Mchars = Substr(yyy,1,At(Chr(13)+Chr(10),yyy))
    If Substr(Mchars,1,1) = '+'
      mchars = Substr(Mchars,2,Len(Mchars)-1)
    Endif
    If Substr(Mchars,1,1) = '+'
      mchars = Substr(Mchars,2,Len(Mchars)-1)
    Endif
    If mline > 1
      ? Rtrim(Ltrim(Mchars))
    Else
      ? Space(4) + Rtrim(Ltrim(Mchars))
    Endif

422
yyy = Substr(yyy, At(chr(13), yyy)+2, Len(yyy)- At(chr(13), yyy))
Mline = mline + 1
Enddo
Else
    If Len(Trim($x81)) > 1
        && Displays Abstract on Screen
        Do Skeleton with 'BROWSING', 'Y', 'BROWSE SCREEN'
        @21,0 Say 'Press ESC key to continue'
        @22,0 Say 'Use PgUp and PgDn to browse note'
        $x81 = Memoedit($x81,6,12,19,78, .f.)
    Endif
Endif

If Popt = 'Y' .or. Popt = 'y'
    Set Print Off
? ?
    @23,0 Say Space(80)
    @23,0 Say 'Continue/Quit Browsing (C/Q) : ' Get Qopt Picture '!
    Read
    If Qopt = 'Q'
        Exit
    Endif
    Set Print on
Else
? ?
    @23,0 Say Space(80)
    @24,0 Say Space(80)
    @23,0 Say 'Continue/Quit Browsing (C/Q) : ' Get Qopt Picture '!
    Read
    If Qopt = 'Q'
        Exit
        Clear
    Endif

? Space(60)
? '**************************************************************'
? Space(60)
? Space(60)
Endif    && Matches Brn range test
Endif    && Matches Depot test
Skip
Enddo

If popt = 'Y' .or. Popt = 'y'
    Set print off
    Eject
Endif
Enddo
RETURN
** Program : Nrep
** Written by K Chetty
** Date 12/31/93
** Aim : Prints NAREM report

Select 1
Use Table1 Index Table1, Ref, Tkey, Mkey, Ukey
Select 2
Use Depot Index Depot
Select 1
Set Procedure to Procfl

Skelvar = ‘N A R E M   R E P O R T S’
Popt = ‘N’
Topflag = .T.
Do While .T.
Do While .T.
   If Topflag
   Else
      Wait       \& Halts screen to view messages
   Endif
   Topflag = .T.
   Popt = ‘N’
   Clear
Skelvar3 = ‘NAREM SCREEN’
Do Skeleton with Skelvar, ‘Y’, Skelvar3
Depotx = Space(20)
Refz = Space(20)
Brn1x = 1
Set order to 1  \& Finds highest Brn
Go Bottom      \& Finds highest Brn
If .Not. Eof()
   Mbrnx = Brn
Else
   Mbrnx = 1
Endif
Brn2x = Mbrnx

@8, 10 Say ‘Depot Code : ’ ^e Depotx
@10,10 Say ‘Reference Number (Prefix e.g. A) : ’ ;
Get Refz Picture ‘!!!!!!!!!!!!!!!!!!!!!’
@12,10 Say ‘Print (Y/N) : ’ Get Popt ‘Picture ’!’
@14,10 Say ‘Brn Range : ’ Get Brn1x
@14,50 Get Brn2x
@16,10 Say ‘To exit leave Depot code and Reference Blank’
@17,10 Say ‘Use Brn range to print selected reports’
Read

If Len(Trim(Depotx)) = 0 .And. Len(Trim(Refz)) = 0
   Return
Endif

424
If Brnx2 < Brnx1
@23.0 Say Space(80)
@23.0 Say 'High Brn less than lower Brn - High Brn reset to lower Brn'
Brnx2 = Brnx1
Endif

Select 1
Set order to 2
Seek Trim(Refz)
If Found()
Else
   @22.0 Say 'No records with Prefix Found, Retry'
   Topflag = .N.
   Wait
Endif

If Len(Trim(Depotx)) = 0
   @21.0 Say 'Depot Code is Mandatory'
   Topflag = .N.
Endif

Dcodex = Upper(Depotx)
Depflag = .T.
Go Top
Do Depotval && Validates Depot code
   If Depflag
      Exit
   Else
      @21.0 Say 'Invalid Depot Code entered'
      Topflag = .N.
   Endif
Enddo

If Popt = 'Y'
   Set Print On
Endif

Refx = Space(20)
Typex = 'M'
Date1x = '000000000'
Date2x = '000000000'
Titlex = Space(1)
Summaryx = Space(1)
Remarksx = Space(1)

Select 1
Set order to 2
Go Top
Seek Upper(Trim(Refz))
Do While .Not. Eof() .and. Upper(Ref) = Upper(Trim(Refz))
   Brntrue = .F.
   If Brn >= Brnx1 .and. Brn <= Brnx2
   Brntrue = .T.
   Endif
   If Upper(Depot) = Upper(Depotx) .and. Narem = nd. Brntrue
     Depotx = Depot
     Sx73 = Space(1)  && Donor Note
     Sx79 = Space(1)  && Access Note
     Sx81 = Space(1)  && Summary
     Sx84 = Space(1)  && Finding Aids
     Sx86 = Space(1)  && Placement with other Collections
     E7 = Space(1)    && Notes in compressed form
     E3 = Space(1)    && Title
     E5 = Space(1)    && Physical description
     Do Uconnem
        && Unstrings first level of Mem field
        Connotex = E7
        If Len(Trim(Connotex))<2
           Connotex = Space(1)
        Endif
        If Len(Trim(Connotex)) > 2
           Do Uconnote
              && Unstrings Notes
           Endif
   Endif

*** Assigning data to variables
   Refx = Ref
   Date1x = Date1
   Date2x = Date2

** Now removing leading zeros from dates
   If substr(Date1x,3,2) = '00'
      Date1x = Substr(Date1x,1,2) + Substr(Date1x,5,4)  && 00 removed if day blank
   Else
      Date1x = Substr(Date1x,3,2) + Substr(Date1x,1,2) + Substr(Date1x,5,4)
   Endif

   If substr(Date2x,3,2) = '00'
      Date2x = Substr(Date2x,1,2) + Substr(Date2x,5,4)  && 00 removed if day blank
   Else
      Date2x = Substr(Date2x,3,2) + Substr(Date2x,1,2) + Substr(Date2x,5,4)
   Endif

   If Len(Date1x) = 6
      If Substr(Date1x,1,2) = '00'
         Date1x = Substr(Date1x,3,4)  && 00 removed if month blank
      Endif
   Endif

   If Len(Date2x) = 6
      If Substr(Date2x,1,2) = '00'

426
Date2x = Substr(Date2x, 3, 4)  && 00 removed if month blank
Endif
Endif

If Len(Date1x) = 4
If Substr(Date1x, 1, 4) = '0000'
Date1x = Space(4)  && Set to blank if no date exists
Endif
Endif

If Len(Date2x) = 4
If Substr(Date2x, 1, 4) = '0000'
Date2x = Space(4)  && Set to blank if no date exists
Endif
Endif

Titlex = Ltrim(Trim(T_art) + ' ' + E3)  && Filing article added to title
Summaryx = Sx81
If Type = 'A'
Typepx = 'Accession'
Else
Typepx = 'Manuscript'  && Default value
Endif

*** Composing Remarks Paragraph
Remarksx = Space(1)
Pd = .F.

If Len(Trim(Sx73)) > 0
Remarksx = Trim(Sx73)  && Donor Note added
Endif

If Len(Trim(E5)) > 0
Remarksx = Remarksx + '; ' + Ltrim(Trim(E5))  && Physical description added
Pd = .T.
Endif

If Len(Trim(Sx86)) > 0
Remarksx = Remarksx + '; ' + Trim(Sx86)  && Placement note added
Endif

If Len(Trim(Sx84)) > 0
Remarksx = Remarksx + '; ' + Trim(Sx84)  && Finding aid note added
Endif

If Len(Trim(Sx79)) > 0
Remarksx = Remarksx + '; ' + Trim(Sx79)  && Access note added
Endif

Remarksx = Ltrim(Trim(Remarksx))
If Len(Remarksx) > 1
If Substr(Remarksx, 1, 1) = '; '  && leading '; ' removed
427
Remarksx = Substr(Remarksx,2,Len(Remarksx)-1)
Endif
Else
Remarksx = Space(1)
Endif

**Statements to Display on screen or Print**

Do Case
Case Popt = 'Y' & Prints entry on paper

*Eject
?
?
?
?
?
'DATA FORM'
?
?
'Depot : ' + Depotx
?'Type : ' + Typex:
?'Reference : ' + Trim(Substr(Refx,1,10)) + Ltrim(Substr(Refx,11,10))
?Y=0
Titlex = 'Title : ' + Titlex
Z = Titlex
Lcnt = 1
Do while Lcnt < 5
Lcntx = Ltrim(Str(Lcnt))
W&Lcntx = Space(0)
Lcnt = Lcnt +1
Enddo

Do Clines with '1', y, z & second level heading formatted
Clne2=y

Lcnt = 1
Do while Lcnt < 5
Lcntx = Ltrim(str(Lcnt))
If Len(Trim(W&Lcntx)) > 0
? W&Lcntx
Endif
Lcnt = Lcnt+1
Enddo
?
?'Starting Date : ' + Date1x
?
?'Ending Date : ' + Date2x
?
Remarksx = 'Remarks : ' + Remarksx
Z= Remarksx
Y=0
Lcnt = 1

428
Do while Lcnt < 5
  Lctx = Ltrim(Str(Lcnt))
  W&Lctx = Space(0)
  Lcnt = Lcnt +1
Enddo

Do Clines with '1',Y,Z  
  & & Formats line to approx. 70 characters
  Lcnt = 1
  Do while Lcnt < 5
    Lctx = Ltrim(str(Lcnt))
    If Len(Trim(W&Lctx)) > 0
      W&Lctx
      Endif
    Lcnt = Lcnt+1
  Enddo

  ?
  ? Substr(Summaryx,1,8) = Space(8)
  ? 'Summary :' + Ltrim(Substr(Summaryx,9, Len(Summaryx)-8))
  Else
    & & If at least 8 leading spaces not found
    ? 'Summary :
  Endif
  ? Summaryx
  & & Correct number of leading spaces is 10
Skip

Case Popt = 'N'  
  & & Displays entry on screen
  Clear
  Do Skeleton With Skelvar, 'Y', 'NAREM SCREEN'
    & @4.0 Say 'Depot : ' + Depotx
    & @5.0 Say 'Type : ' + Typex
    & @6.0 Say 'Reference : ' + Trim(Substr(Refx,1,10)) +
      Ltrim(Substr(Refx,11,10))
    & @7.0 Say 'Title : ' + Titlex
    & @11.0 Say 'Starting Date : ' + Date1x
    & @12.0 Say 'Ending Date : ' + Date2x
    If Len(Remarksx) < 254
      & @13.0 Say 'Remarks : ' + Remarksx
      Wait
    Else
      Wait
      Clear
      Do Skeleton with Skelvar, 'Y', 'NAREM SCREEN'
        & @4.0 Say 'Remarks : ' + Remarksx
        Wait
    Endif

  If Len(Trim(summaryx)) >0
    Clear
    Do Skeleton with Skelvar, 'Y', 'NAREM SCREEN'
      @5.0 Say 'Summary :'
      @20.0 Say Space(80)
    If Clipper

429
@22,0 Say 'Press ESC key to Continue'
Summaryx = Memoedit(Summaryx, 6,12,19,78,.F.)
Else
  ? Summaryx
  Endif
  Endif
Skip
Endcase

If Popt = 'Y'
  
  Set Print off
  
  Xopt = 'C'
  @23,0 Say Space(80)
  @23,0 Say 'Continue/Quit Print? ' Get Xopt Picture '!'
  Read
  If Xopt = 'Q'
    Exit
  Endif
  Set Print on
Else
  Xopt = 'C'
  @23,0 Say Space(80)
  @23,0 Say 'Continue/Quit Browsing? ' Get Xopt Picture '!'
  Read
  If Xopt = 'Q'
    Exit
  Endif
Endif

Else
  & Skips if selection criteria, depot, brntrue and Narem not met
Endif
Enddo

If Popt = 'Y'
  Eject
  Set Print Off
Endif
Enddo
RETURN
**Program: Reindex**
**Written by K Chetty**
**Date 12/30/93**
**Program reindexes Table1 and Depot Files**
**Option 3 of menu sorts headings for AACR2R**
**The master control definitions for each index exist here**

Close All
Set Procedure to Procfl
Clear
Set Talk Off
Mainx = ''
Skelvar = 'REINDEXING OPTIONS'

Do While .T.
    Do While .T.
        Clear
        @ 4,0 Say ''
        Text
            1. Reindex Depot File
            2. Reindex Table1
            3. Reindex Table1 (Sort Headings)
            0. Exit
        Endtext
        Do Skeleton with Skelvar, 'N', 'REINDEXING SCREEN'
        @18,0 Say 'Select Option: ' Get Ropt Picture '!' Read
        If Ropt $ '0123'
        Exit
        Endif
    Enddo

Do Case
    Case Ropt = '1'
        Select 2
        Use Depot
        ?'Indexing Depot file'
        Index on Upper(Depcode) to Depot
        Use Depot Index Depot
        ?'Depot file Reindexed successfully'
    Case Ropt = '2'
        ?'Indexing Table1'
        Select 1
        Use Table1
        Index on Brn to Table1

431
Index on Upper(Substr(Ref,1,10) + Substr(Ref,11,10))) to Ref
Index on Tkey to Tkey
Index on Mtype+Mkey to Mtkey
Index on Utype + Ukey to Ukey
Use Table1 Index Table1, Ref, Tkey, Mtkey, Ukey
? 'Table1 indexed successfully except for Main Entry Heading Index'
? 'Use Option 3 of Reindexing menu to sort on Headings'

Case Ropt = '3'
Select 1
Use Table1
** Set up dummy index entries using Maindex = ', so that
** the index is fully index thereafter it can be linked to Table1
Maindex = ',
Index on Upper(Msort) to Maindex
Use Table1 Index Maindex
Set order to 0
? 'Indexing Headings'

** Three cases for Utype exist. Utype = U, T or Blank
** For Utype = ', Mtype could be A,P, or C or Blank'

Go Top
Do while .not. EOF()
  Brnx = Brn
  Mainx = Space(1)
  Mainy = Space(1)
  Mkeyx = Mkey
  Mttype = Mtype
  Utyper = Utype
  Ukeyx = Ukey
  Tkeyx = Tkey

  If Clipper
    Memx = Mem
  Endif

  Connotex = Space(1)
  Cmmemx = Space(1)
  E1 = Space(1)     && Main Entry Heading
  E2 = Space(1)     && Uniform Title
  E3 = Space(1)     && Title
  Do Ucommem     && Unstrings mem field into E1, E2, E3
    Uall = Space(1) && Utility Variable - see below
    Uell = Space(1) && Holds Uniform Entry Element
    Uell2 = Space(1) && Holds Uniform Additions
  Enddo

  ** STRIPPING UNIFORM TITLES INTO Uell, Uell2
  Uall = E2
  If Len(Trim(Uall)) > 1    && Uniform title exists

432
Uall = Trim(Uall)
Uall = Uall + Space(100-Len(Uall))

** Removing leading delimiter **
Uall = Substr(Uall,2,Len(Uall)-1)
E11 = Substr(Uall,1,At(Chr(181), Uall)-1)
Ue11 = E11 + Space(100-Len(E11))
E12 = Substr(Uall, At(Chr(181), Uall)+1, Len(Uall)-
At(Chr(181), Uall)-2)
E12 = Trim(Trim(E12))
E12 = Substr(E12,1, Len(E12)-3) && Last 3 characters removed
If Len(E12) < 31
Ue12 = E12 + Space(30-Len(E12))
Else
Ue12 = Substr(E12,1,30)
Endif
Else
Ue11 = Space(100)
Ue12 = Space(30).
Endif

** Stripping Main Heading Elements **
E11 = Space(100)
E12 = Space(50)
E13 = Space(30)
E14 = Space(1)
E15 = Space(1)
E16 = Space(1)
If Len(Trim(E1))>1
Mainy = E1
Do Ucommain && else takes on initial values
Endif

Titlex = Upper(E3)
If '/ ' $ Titlex
Titlex = Substr(Titl,1, At('/ ', Titlex)-1)
Endif
Titlex = Trim(Titlex) + Space(100)
Titlex = Substr(Titlex,1,100)
If Len(E12) > 29
E12 = Substr(E12,1,30)
Endif

If Len(E13) > 29
E13 = Substr(E13,1,30)
Endif

Do Case
Case Utype = 'U' && Uniform Title Exists
If Mtype = 'A' && with Author
If Len(Uell) <= 50

433
Else
Uell = Substr(Uell,1,50)
Endif
Mindex = E11 + Space(100-Len(E11)) + E12 + Space(30-Len(E12)) ;
+ Uell + Space(50- Len(Uell))+ Uell2 + Space(30-Len(Uell2));
+ Substr(Titlex,1,40)
Endif

If Mtype = 'P' & & Place exists
If Len(Uell) <= 50
Else
Uell = Substr(Uell,1,50)
Endif
Mindex = E11 + Space(100-Len(E11)) + E13 + Space(30-Len(E13));
+ Uell + Space(50- Len(Uell))+ Uell2 + Space(30-Len(Uell2));
+ Substr(Titlex,1,40)
Endif

If Mtype = 'C' & & Corporate Body exists
If Len(Uell) <= 50
Else
Uell = Substr(Uell,1,50)
Endif
Mindex = E11 + Space(100-Len(E11)) + E13 + Space(30-Len(E13));
+ Uell + Space(50- Len(Uell))+ Uell2 + Space(30-Len(Uell2));
+ Substr(Titlex,1,40)
Endif

If Mtype = '-' & & No Place, Name, or Corporate Body
Mindex = Uell + Space(100-Len(Uell)) + Uell2;
+ Space(30-Len(Uell2)) + Space(80) + Substr(Titlex,1,40)
Endif

Case Utype = 'T' & & Title Main Entry
Mindex = Titlex + Space(100-Len(Titlex)) +;
Space(110) + Substr(Titlex,1,40)

Case Utype = '-' & & No Uniform Title
If Mtype = 'A' & & with only Author
Mindex = E11 + Space(100-Len(E11)) + E12 + Space(30-Len(E12));
+ Space(80) + Substr(Titlex,1,40)
Endif

If Mtype = 'P'
Mindex = E11 + Space(100-Len(E11))+ E13 ;
+ Space(30-Len(E13)) + Space(80)+ Substr(Titlex,1,40)
Endif

If Mtype = 'C'
Mindex = E11 + Space(100-Len(E11)) ;
+ E13 + Space(50- Len(E13)-20) + Space(80)+ Substr(Titlex,1,40)
Endif

434
** No Main Entry Headings
If Mtype = '/'
    Mainindex = Space(210) + Substr(Titlex,1,40)
Endif
Endcase

Replace MSORT with Upper(Mainindex)
Skip
Enddo

? 'Indexing Complete'
Wait

Case Ropt = 'O'
Exit
Endcase
Enddo
Enddo
RETURN
** Depaut.prg
** Date 12/31/93
** Written by K Chetty
** Aim: to create, update, delete and
** list Depot Codes and descriptions

Close All
Select 2
Use Depot Index Depot
Set Procedure to Procfl
Depopt = '0'

Do While .t.
Clear
Skelvar = 'DEPOT FILE MAINTENANCE'
Do Skeleton with Skelvar, 'Y', 'DEPOT SCREEN'

Do while .t.
Clear
@4,0 say '','
Text

1. Create Code and Description
2. Update Description
3. Update Code
4. Delete Code
5. List codes and descriptions
0. Exit
Endtext
Do Skeleton with Skelvar, 'N', 'DEPOT SCREEN'
@14,5 Say 'Select Option : ' Get Depopt Picture '!' Read
If Depopt $'012345'
Exit
Endif
Enddo

If Depopt = '0'
Exit
Endif

Do Case
Case Depopt = '1'
Do While .T.
Clear
Skelvar = 'DEPOT INPUT'
Do Skeleton with Skelvar, 'Y', 'DEPOT INPUT SCREEN'
Depcodex = Space(20)
Depdescx = Space(50)
@5,5 Say 'Enter Depot Code : ' Get Depcodex
@7,5 Say ' Description : ' Get Depdescx

436
Read
If Len(Trim(Depcodex)) = 0
Exit
Endif
Go Top
Seek Upper(Depcodex)
If Found()
@9,5 Say 'Code Already exists'
@11,5 Say 'Code is' + Depcodex
@13,5 Say 'Description is' + Depdesc
Else
@9,5 Say 'Creating new record'
Append Blank
Replace Depcode with Depcodex
Replace Depdesc with Depdescx
Endif
Wait
Enddo

Case Depopt = '2'
Do While .T.
Clear
Skelvar = 'DEPOT DESC UPDATE'
Do Skeleton with Skelvar, 'Y', 'DEPOT DESCRIPTION UPDATE SCREEN'
Depcodex = Space(20)
Depdescx = Space(50)
@5,3 Say 'Enter Depot Code : ' Get Depcodex
Read
If Len(Trim(Depcodex)) = 0
Exit
Endif
Go Top
Seek Upper(Depcodex)
If Found()
    Depdescx = Depdesc
    @11,3 Say 'Enter new Description : ' Get Depdescx
    Read
    Replace Depdesc with Depdescx
Else
    @9,3 Say 'Code does not exist'
Endif
Wait
Enddo

Case Depopt = '3'
Do While .T.
Clear
Skelvar = 'DEPOT CODE UPDATE'
Do Skeleton with Skelvar, 'Y', 'DEPOT CODE UPDATE SCREEN'
Depcodex = Space(20)
Depdescx = Space(50)
@5,5 Say 'Enter Depot Code : ' Get Depcodex
Read
If Len(Trim(Depcodex)) = 0
Exit
Endif
Go Top
Seek Upper(Depcodex)
If Found()
    Depcodex2 = Depcodex
    @10, 5 Say 'Description is : ' + Depdesc
    @11, 5 Say 'Enter new code : ' Get Depcodex2
    Read
    Go Top
Seek Upper(Depcodex2)
    If Found()
        @13, 5 Say 'Code exists, code not changed'
    Else
    Go Top
    Seek Upper(Depcodex)
    If Found()
        Replace Depcode with Depcodex2
    Else
        @14, 5 Say 'Unable to find Depot Record, update failed'
    Endif
Endif
Else
    @9, 5 Say 'Code does not exist'
Endif
Wait
Enddo

Case Depopt = '4'
Do While .T.
    Clear
    Skelvar = 'DEPOT CODE DELETION'
    Do Skeleton with Skelvar, 'Y', 'DEPOT CODE DELETION SCREEN'
    Depcodex = Space(20)
    Depdesc = Space(50)
    @5, 5 Say 'Enter Depot Code : ' Get Depcodex
    Read
    If Len(Trim(Depcodex)) = 0
       Exit
    Endif
Endif

Go Top
Seek Upper(Depcodex)
If Found()
    If Deleted()
        Dopt2 = 'Y'
        @10, 5 Say 'Code already deleted'
        @11, 5 Say 'Description is : ' + Depdesc
        @12, 5 Say 'Undelete/Recall Item (Y/N) : ' Get Dopt2 Picture '!'
        Read

        438
If Dep2 = 'Y'
   @13,5 Say 'Code UNDELETED/RECALLED'
   Recall
Endif
Else
   Delete
   @11,5 Say 'Deletion successful'
Endif
Else
   @12,5 Say 'Code not found, not deleted'
Endif
Wait
Enddo

Case Depopt = '5'
   Clear
   Lne = 7
   Go Top
   Skelvar = 'DEPOT LISTING'
   Do While .not. Eof()
      Do Skeleton with Skelvar, 'N', 'DEPOT LIST SCREEN'
      @6,0 Say 'CODE
      @Lne,0 Say Depcode
      @Lne,20 Say Depdesc
      Skip
      Lne = Lne + 1
      If Lne = 14
         Wait
         Clear
         Lne = 7
      Endif
   Enddo
   Wait
Endcase
Enddo
Close All
RETURN
APPENDIX D

The Procedures called by programs in Appendix C are stored in Procfl which is listed here. The procedures are listed alphabetically within the procedure file, Procfl.
**Procedure file : Procfl
** Written by K Chetty
** Date : 12/31/93
**************************

Procedure BrowNote
** Retrieves ALL notes entered for an entry during input or update.
** Procedure invoked by placing a '?' in when selecting a note
** in the Note Procedure

Clear
P2 = 65
Do While P2 > 64 .and. P2 < 89
Rx = Ltrim(Str(P2))
Sx&Rx = Trim(Sx&Rx)

If Len(Trim(Sx&Rx)) > 0
   Do Skeleton with 'BROWSING NOTES', 'y', 'BROWSE SCREEN'
      If Clipper
         @21,0 Say Space(80)
         @21,0 Say 'Press ESC key to continue'
         @22,0 Say Space(80)
         @22,0 Say 'Use PgUp or PgDn to view more than one page'
         Sx&Rx = Memoedit(Sx&Rx,6,12,19,78,.F.)
      Else
         ? Sx&Rx
         @21,0 Say 'Press ESC key to continue'
         Wait
         Endif
   Endif
P2 = P2 + 1
Enddo
RETURN

*****

Procedure Centre
Parameters Lineno, Skelvar
** Procedure centres heading

Skelvar = Ltrim(Skelvar)
Skelvar = Rtrim(Skelvar)
If Len(Skelvar) > 79
   Skelvar = Substr(Skelvar,1,79)
Endif
Cntrc = (80 - Len(Skelvar))/2
@ Lineno,Cntrc Say Skelvar
RETURN

*****
Procedure Clines
** Formats the number of lines for AACR2R and ISBD Output

Parameters x, y, z
Private Tflag

Do case
  Case x = '1'
    Do Case
      Case Len(z) <= 72
        lcnt = 1
        lcntx = Ltrim(Str(lcnt))
        W&lcntx = z
        y = 1
        Return
      Endcase
    Case Len(z) > 72
      Lastl = .F.
      lcnt = 1
      y = 0
    Endcase
    Do while Len(z) > 0
      If Len(z) <= 72
        Lastl = .T.
        lcntx = Ltrim(Str(lcnt))
        W&lcntx = z
        y = y + 1
        Exit
      Endif
      zz = Substr(z, 1, 72)
      Atcnt = 72
      Watfind = .F.
      Do while atcnt > 49
        wat = Substr(zz, atcnt, 1)
        If wat = ''
          Watfind = .T.
          zz = Substr(zz, 1, atcnt)
        Endif
        lcntx = Ltrim(Str(lcnt))
        W&lcntx = zz
        Y = Y + 1
        z = Substr(z, atcnt + 1, Len(z) - atcnt + 1)
      Enddo
      lcnt = lcnt + 1
      Exit
    Enddo
  Endcase
  Enddefault
Endcase
If Watfind
  Else
    lcntx = Ltrim(Strip(lcnt))
    W&lcntx = zz
    z = substr(z, 73, Len(z)-73)
    lcnt = lcnt+1
    y = y+1
  Endif
Enddo
Endcase
Endcase
RETURN

******

Procedure Conmain
Parameters E11, E12, E13, E14, E15, E16
** Strings 6 elements of the Main Entry Heading.
** Uconmain performs the reverse of this procedure.
Mainy = Chr(181) + Trim(E11) + Chr(181) + Trim(E12) + Chr(181) + Trim(E13) +;
Chr(181) + Trim(E14) + Chr(181) + Trim(E15) + Chr(181) + Trim(E16) + Chr(181)
RETURN

*****

Procedure Conmem
** Concatenates Mem field (level 1) for 7 subfields, using delimiter Chr(180)
** Fields are in order Main, Uniform, Title, Edition, Physical description,
** Date and Notes.
Conmemx = Chr(180) + Trim(Mainx) + Chr(180) + Trim(Uniformx) + Chr(180) +;
Trim(Titlex) + Chr(180) + Trim(Edx) + Chr(180) + Trim(Phdescx) + Chr(180) +;
Trim(datx) + Chr(180) + Trim(Connotex) + Chr(180)
RETURN

*****

Procedure Connote
** Concatenates the notes into one string using delimiter, Asc(181)

Connotex = Space(1)
P2 = 65
Do While P2 > 64 .and. P2 < 89
  Ax = Ltrim(Strip(P2))
  If Len(Trim(Sx&Ax)) > 1
    Sx&Ax = Chr(&Ax) + Trim(Sx&Ax) && Stores the Indicator Chr(&Ax) with note
    Connotex = Connotex + Chr(181) + Sx&Ax
  Endif
  P2 = P2 + 1
Enddo
Connotex = Ltrim(Connotex) + Chr(181) && Extra delimiter added at end
RETURN

443
Procedure Corp
** Inputs and Updates Corporate Headings

Skelvar = 'CORPORATE BODIES: INPUT/UPDATE'
If Len(Trim(ElementIx)) > 0 .or. Progname = 'UPROG'
   E11 = Space(1)
   E13 = Space(1)
   Mainy = Mainx
   Do Uconmain
      ElementIx = E11 + Space(100-Len(E11))
      Addsx = E13 + Space(50-Len(E13))
   Else
      ElementIx = Space(100)
      Addsx = Space(50)
   Endif

Flag1 = 'T'
Do While .T.
   If Flag1 = 'N'
      wait
   Endif

   Do Skeleton with Skelvar, 'Y', 'CORP SCREEN'
   Qopt = 'N'
   Flag1 = 'T'
   @10,5 Say 'Corporate Body: including Subdivisions and parts where applicable'
   @12,23 Get ElementIx
   @16,5 Say 'Additions: ' Get Addsx
   Read

   If Len(Trim(ElementIx)) = 0
      @22,0 Say 'First Element is Mandatory, Quit (Y/N): ' Get Qopt Picture '!'
      Read
      If Qopt = 'Y'
         ElementIx = Space(100)
         Addsx = Space(50)
         Mtypex = '/'
         Exit
      Else
         Flag1 = 'N'
      Endif
   Endif

If Substr(ElementIx,1,1) = '?'
   @21,0 Say Space(80)
   @22,0 Say Space(80)
   @21,0 Say 'Enter Main and subordinate bodies. Separate with full stops'
   @22,0 Say 'E.g. Stanford University. Department of Civil Engineering'
   Flag1 = 'N'

444
Endif

If Substr(Addsx,1,1) = '?'
@21,0 say Space(80)
@22,0 Say Space(80)
@21,0 Say 'Enter additions e.g. (Ship) as in Bounty (Ship)'
@22,0 Say 'Conference additions e.g. (1st :1919 Feb. 11-15 : New Orleans, La.)'
Flag1 = 'N'
Endif

If Flag1 = 'T'
Exit
Endif
Enddo

Keyx = Space(6)
Elx = Element1x
Do wcomp2 with Elx, Keyx && Compose Mkey for Corporate Bodies

Mkeyx = Upper(keyx)
If Progname = 'IPROG' .and. Len(Trim(Mkeyx)) > 1
Do Mainval with Mtype, Mkeyx && Validates Corporate Bodies
Endif

** Constructs Storage format for Main Entry Heading
Mainx = Space(1)
Mainy = Space(1)
Do Conmain with Element1x, ",", Addsx, ",", ",", "
Mainx = Mainy
RETURN

******
Procedure Depot
*Aim = Used during input and update to create and view Depot codes

Select 2
Use Depot Index Depot
Clear
Skelvar = 'DEPOT CODE : CREATE/UPDATE'
Do Skeleton with Skelvar, 'Y', 'DEPOT SCREEN'

Qty="C"

If Progname = 'IPROG'
Dcodex = "?" + Space(19)
Endif
Copt = "1"
Ddescx = Space(50)
Cnt = 1

Do While .T.
Clear
Do Skeleton with Skelvar, 'y', 'DEPOT SCREEN'
@14,24 SAY "SELECT OPTION" GET copt
@8,24 Say '1. View and Select Depot Codes'
@10,24 Say '2. Add a New Depot Code'
@12,24 Say '3. Exit Option'
Read
If Copt = '3'
Exit
Endif

Do Case
Case copt = "1"
Clear
Do Skeleton with Skelvar, 'y', 'DEPOT SCREEN'
@5,4 Say "Depot Code, To exit press ENTER :
@5,40 Get Dcodex
Read
Dcodex = Upper(Substr(Dcodex, 1,1)) + Substr(Dcodex,2,19)
Dcodex = Rtrim(Dcodex)
Cnt = 8
@6,4 Say "IF CODE DOES NOT MATCH, SELECT OPTION 2 OF PREVIOUS MENU If DESIRED"
@7,4 Say "CODE DESCRIPTION"
Do While .T.
If Substr(Dcodex,1,1) = '?' && Lists all entries
Go Top
Do While .Not. EOF()
@cnt,4 Say Depcode
@ cnt,28 Say Depdesc
Qty = 'C'

446
cnt = cnt + 1
if cnt >= 16
  @2,4 say space(50)
@19,4 say 'Quit/Continue searching (Q/C) : ' GET Qty PICTURE '
read
if Qty = 'Q'
exit
endif
cnt = 8
endif
skip
enddo
else
  goto top
  seek upper(dcodez)
do while .not. eof() .and. upper(rtrim(depcde)) = upper(dcodez)
    @cnt,4 say depcode
    @ cnt,28 say depdesc
    qty = 'C'
    cnt = cnt + 1
    if cnt >= 16
      @2,4 say space(50)
      @19,4 say 'Quit/Continue searching (Q/C) : ' GET Qty PICTURE '
read
if Qty = 'Q'
exit
endif
  cnt = 8
endif
  skip
enddo
endif

35,40 get dcode
read
exit
dendo
do depotval
exit

case copt = '2'
  clear
do skeleton with skelvar, 'y', 'DEPOT SCREEN'
do while .t.
  @12,5 say 'Depot Code : ' get dcode
  @14,5 say 'Description : ' get ddesc
read
  dcode = upper(substr(dcode, 1, 1)) + substr(dcode, 2, 19)
  if len(trim(dcode)) = 0
    clear

447
CODE

@14,4 Say 'DEPOT CODE IS MANDATORY, USE OPTION ONE TO SELECT

Wait
Clear
Exit
Endif
Go Top
Tflag = .T.
Seek Upper(Trim(Dcodex))
Do While .Not. Eof() .And. Upper(Deppcode) = Upper(Dcodex)
   Do Skeleton with Skelvar, 'y', 'DEPOT SCREEN'
   @12,14 Say 'Code already Exists'
   @12,35 Say Deppcode
   @14,14 Say 'Description :
   @14,35 Say Depdesc
   Wait
   Tflag = .N.
   Exit
   Skip
Enddo

If Tflag
   Do Skeleton with Skelvar, 'y', 'DEPOT SCREEN'
   @16,4 Say 'CREATING NEW AUTHORITY RECORD'
   @18,4 Say 'Newly Created Code is accepted'
   Append Blank
   Replace Depcode With Dcodex
   Replace Depdesc With Ddescx
   Wait
   Clear
Endif
Exit
Enddo
Endcase
Enddo
Select 1
RETURN
******
Procedure Depotval
** Validates a depot code
Select 2
Use Depot Index Depot
Go Top
Set Exact on
Seek Upper(Dcodex)

If Found()
Else
Depflag = .N.
Endif

If Eof()
Depflag = .N.
Endif

Set Exact Off
Select 1
RETURN
******

Procedure ISBDB
** Procedure produces a brief description which is used during
** title validation and update. The procedure outputs the ISBD
** (excluding notes ) and the BRN, Source, Reference, Class number.
** In the Update menu, a brief record is presented before the
** segments of the entry (e.g. title, main entry heading etc.)
** are presented for update.

E3 = ' '
E4 = ' '
E5 = ' '
E6 = ' '
Do Uconmem

** Details extracted from current record in Table1
Classz = Class
Brzn = Brn
Lin1 = Space(80)
Lin1 = 'BRN : ' + Str(Brn) + Space(10-Len(Str(Brn))) + Space(14) + 'Ref : ';
   + Trim( Substr(Ref,1,10)) + Substr(Ref,,11,10) +;
   Space(43- Len(Trim( Substr(Ref,1,10)) + Substr(Ref,,11,10))) + 'Source : ' +
   Depot
Lin1 = Lin1 + Space(20- Len(Depot)) + ' Class : ' + Class
Lin1 = Lin1 + Space(160- Len(Lin1)) && Two lines reserved for BRN,  
   Fsds = ' . - ' && Classification etc.
Isbdx = Trim (Lin1)

449
** Depending on whether field is empty, ISbdx is appended to; accordingly
     ISbdx = ISbdx + Space(160 - Len(ISbdx)) + Ltrim(Trim(T_art) + ' ' + Trim(E3))
     & Title
     If Len(Trim(E4)) > 0
     ISbdx = ISbdx + Fsds + Trim(E4)
     & Edition
     Endif
     If Len(Trim(E6)) > 0
     ISbdx = ISbdx + Fsds + Trim(E6)
     & Date
     Endif
     If Len(Trim(E5)) > 0
     ISbdx = ISbdx + Fsds + Trim(E5)
     & Physical description
     Endif
     RETURN

*****

Procedure Ititle
** Procedure inputs and Updates titles and filing article
** and Statement of Responsibility

Skelvar = 'I N P U T T I T L E S'
Titlefl = .T.
Do While .T.
   If Titlefl
      Else
         wait & Pauses on errors and messages
      Endif
      Titlefl = .T.
      Clear

** Next 7 lines initialises variables
** If called from Uprogs existing data used and fields set to
** Maximum lengths
     If Progname = 'UPROG'
        Skelvar = 'U P D A T E T I T L E S , '
        T_Artx = T_artx + Space(10-Len(T_artx))
     Endif
     If Clipper
        Titlex = Titlex + Space(500-Len(Titlex))
     Else
        Titlex = Titlex + Space(254-Len(Titlex))
     Endif
     Else
        T_artx = Space(10)
     Endif
     If Clipper
        Titlex = Space(500)
     Else
        Titlex = Space(254)
     Endif
     Endif

Do Skeleton with Skelvar, 'Y', 'TITLE SCREEN' & Sets screen layout
@5,10 Say 'Filing Article : ' Get T_artx

450
@7,10 Say 'Title and Statement of Responsibility :'
@8,10 Get Title
Read
    If Trim(T_artx) = '?' && Help on filing Article
@21,10 Say Space(80)
@21,10 Say 'Enter a filing article e.g. A, An, The'
    T_artx = Space(10)
    Titlef1 = .N.
Endif

    If Len(Trim(Titlex)) = 0
        If Progname = 'UPROG'
            @21,10 Say 'Title is Mandatory : Re-Enter'
            Wait
            Titlef1 = .N.
        Else
            @21,10 Say 'Title is Mandatory : Exiting Input Program'
            Wait
            Return
        Endif
    Endif

    If Substr(Titlex,1,1) = '?'
@22,10 Say Space(80)
@22,10 Say 'Enter a title and one or more statements of responsibility'
@23,10 Say Space(80)
@23,10 Say "Using the separators '/' and ',' respectively"
    Titlef1 = .N.
Endif

    If Len(Trim(Titlex)) > 0 .and. Titlef1
        Exit
    Endif
Enddo
RETURN

****
Procedure Main
** Main procedure sets up the variables for the Main Entry Heading.
** Two sets of variables, one for the heading and other for Uniform titles.
** Calls to procedures Name, Place, Corp and Uniform are made from here.
** Procedure ensures that only one Name, Place or Corporate body is entered

Clear
Mvar = '1'
If Progname = 'UPROG'
  Else
    Mtypex = ' '
    Mkeyx = Space(6)
    Mainx = Space(1)
    Ukeyx = Space(6)
    Utypex = Space(1)
    Uniformx = Space(1)
  Endif
Uflag = .T.
Element1x = Space(1)
Element2x = Space(1)
Element3x = Space(1)
Addsx = Space(1)
Neblix = Space(1)
Britishx = ' '
Element1lx = Space(1)
Addslxlx = Space(1)

Do While .T.
  Do While .T.
    Clear
    Mvar = '0'
    Uflag = .T.
    @4, 0 Say "" 
    Text
      1. Names
      2. Place
      3. Corporate Bodies
      4. Uniform Titles
      5. Title
      6. None of the Above (No Main Entry Heading)
      0. Exit (After Selection of Main Entry Headings)
    Endtext
    Skelvar = 'MAIN ENTRY HEADING S'
    Do skeleton with skelvar, 'N', 'HEADING SCREEN'
    @15, 50 Say 'BRN = ' + Str(Brnx)
    ** Next line displays options A,P,C,U,T on screen, if selected already
    @17, 10 Say ' Currently selected options are : ' + Mtypex + Utypex
    @15, 10 Say 'Select Main Entry Heading : ' Get Mvar
    Read

452
** Following lines test if title main entry heading entered with a 
** Name, Place, Corporate, or Uniform title, an error message is given. 
** In case of conflict the title is deleted as main entry heading 
** Option 6 must be used to initialise headings 

    If Utypex = 'T'
    If Mtypex = 'A' .or. Mtypex = 'P' .or. Mtypex = 'C' .or. Mtypex = 'U'
    Clear
    @12,12 Say 'Choose either title or one of Author, Place, Corporate Body'
    @14,12 Say 'Combine Uniform Title with one of Author, Place or Corporate Body'
    @16,12 Say 'Deleting title as main entry heading'
    Utypex =
    Uflag = .N.
    wait
    Endif
Endif

** Following statements ensures that if author chosen, place 
** or corporate heading are not allowed
If Mtypex = 'A' .and. (Mvar = '2' .or. Mvar = '3')
    @18,0 Say 'Invalid Option Chosen, Select Option 1 or Option 6 to Clear existing headings'
    wait
    Uflag = .N.
Endif

** Following statements ensures that if corporate body chosen, place 
** or author are not allowed
If Mtypex = 'C' .and. (Mvar = '1' .or. Mvar = '2')
    @18, 0 Say 'Invalid Option chosen, Select Option 3 or Option 6 to Clear existing headings'
    wait
    Uflag = .N.
Endif

** Following statements ensures that if place chosen, 
** corporate body or author are not allowed
If Mtypex = 'P' .and. (Mvar = '1' .or. Mvar = '3')
    @18, 0 say 'Invalid option chosen, Select Option 2 or Option 6 to Clear existing headings'
    wait
    Uflag = .N.
Endif

    If Mvar $'1234560?' .and. Uflag
    Exit
    Endif
Enddo
Clear

Do Case
Case Mvar = '1'
Skelvar = 'NAME HEADINGS'
Do Skeleton with Skelvar, 'y', 'NAME SCREEN'
Mtypex = 'A'
Do Names

Case Mvar = '2'
Skelvar = 'PLACE NAMES'
Do Skeleton with Skelvar, 'y', 'PLACE SCREEN'
Mtypex = 'P'
Do Place

Case Mvar = '3'
Skelvar = 'CORPORATE BODIES'
Do Skeleton with Skelvar, 'y', 'CORP SCREEN'
Mtypex = 'C'
Do Corp

Case Mvar = '4'
Skelvar = 'UNIFORM HEADINGS'
Do Skeleton with Skelvar, 'y', 'UNIFORM SCREEN'
Utypex = 'U'
Do Uniform

Case Mvar = '5'
@21.0 Say Space(80)
Do skeleton with Skelvar, 'y', 'TITLE SELECTION'
@12.0 Say 'Title accepted as Main Entry Heading'
Wait
Utypex = 'T'
Ukeyx = Space(6)
Uniformx = Space(1)
Mkeyx = Space(1)
Mtypex = Space(1)
Mainx = Space(1)

Case Mvar = '6'
Mkeyx = Space(1)
Mtypex = Space(1)
Mainx = Space(1)
Utypex = 'U'
Uniformx = Space(1)
Ukeyx = Space(1)
Element1x = Space(1)
Element2x = Space(1)
Element3x = Space(1)
Addsx = Space(1)
Britishx = 'N'
Nobilx = Space(1)
Element1lx = Space(1)
Addslx = Space(1)
Do Skeleton with Skelvar, 'y', 'HEADING SCREEN'
@12.6 Say 'All of Place, Name, Corporate'
@14.6 Say 'Uniform Titles have been cleared'
Wait

Case Mvar = '0'
exit
Case Mvar = 'Q'
Clear
@4,0 Say ""
Text

Names, Places and Corporate bodies are entered in accordance with
the AACR2R. The Entry or first element of each screen is mandatory
while the other elements are mandatory, if applicable. ONLY one of
these three types of headings may be included in the heading and
combined with a uniform title if present, (option 4). Option 5
allows for a title main entry, while option 6 clears all
main entry headings in the system allowing for an ALTERNATE ENTRY
approach or for cases when no heading is defined. Option 6
allows the operator to EXIT, when all headings have been
chosen.
Endtext
Do Skeleton with Skelvar, 'N',''
wait
Endcase
Enddo

** Ascertaining structure and content of Msort
** Three cases for Utype exist. Utype = U, T or Blank
** For Utype = 'U', Mtype could be A,P, or C or Blank

E1 = Mainx && Main Entry Heading
E2 = Uniformx && Uniform Title
E3 = Titlex && Title

Uall = Space(1) && Utility Variable - see below
Uel1 = Space(1) && Holds Uniform Entry Element
Uel2 = Space(1) && Holds Uniform Additions

** STRIPPING UNIFORM TITLES INTO Uel1, Uel2

Uall = E2
If Len(Trim(Uall)) > 1 && Uniform title exists
Uall = Trim(Uall)
Uall = Uall + Space(100-Len(Uall))

* Removing leading delimiter
Uall = Substr(Uall,2,Len(Uall)-1)
E11 = Substr(Uall,1,At(Chr(181), Uall)-1)
Uel1 = E11 + Space(100-Len(E11))
E12 = Substr(Uall, At(Chr(181), Uall)+1, Len(Uall)-
At(Chr(181), Uall)-2)
E12 = ltrim(Trim(E12))
E12 = Substr(E12,1, Len(E12)-3) && Last delimiter removed
If Len(E12) < 31
Uel2 = E12 + Space(30-Len(E12))
Else
Uel2 = Substr(E12,1,30)
Endif

455
Else
  Ue11 = Space(100)
  Ue12 = Space(30)
Endif

** Stripping Main Heading Elements
E11 = Space(100)
E12 = Space(50)
E13 = Space(30)
E14 = Space(1)
E15 = Space(1)
E16 = Space(1)
If Len(Trim(E1))>1
  Mainy = E1
  Do Ucomain
  Endif
  & & else takes on initial values
E3 = Upper(E3).
If ' / ' $ E3
  E3 = Substr(E3,1,At(' / ',E3)-1)
Endif
E3 = Trim(E3) + Space(100)
E3 = Substr(E3,1,100)

If Len(E12) > 29
  E12 = Substr(E12,1,30)
Endif

If Len(E13) > 29
  E13 = Substr(E13,1,30)
Endif

Mainindex = ' '
Do Case
  Case Utypex = 'U' & & Uniform Title Exists
    If Mtypex = 'A' & & with Author
      If Len(Ue11) <= 50
        Else
          Ue11 = Substr(Ue11,1,50)
        Endif
        Mainindex = E11 + Space(100-Len(E11)) + E12 + Space(30-Len(E12)) +
        + Ue11 + Space(50-Len(Ue11)) + Ue12 + Space(30-Len(Ue12)) +
        + Substr(E3,1,40)
        Endif
      If Mtypex = 'P' & & Place exists
        If Len(Ue11) <= 50
          Else
            Ue11 = Substr(Ue11,1,50)
          Endif
        Mainindex = E11 + Space(100-Len(E11)) + E13 + Space(30-Len(E13)) +
        + Ue11 + Space(50-Len(Ue11)) + Ue12 + Space(30-Len(Ue12));
+ Substr(E3,1,40)
Endif

If Mtypep = 'C' && Corporate Body exists
  If Len(Uell) <= 50
    Else
      Uell = Substr(Uell,1,50)
    Endif
    Mainindex = E11 + Space(100-Len(E11)) + E13 + Space(30-Len(E13));
    + Uell + Space(50-Len(Uell)) + Uell2 + Space(30-Len(Uell2));
    + Substr(E3,1,40)
  Endif

If Mtypep = ' ' && No Place, Name, or Corporate Body
  Mainindex = Uell + Space(100-Len(Uell)) + Uell2;
  + Space(30-Len(Uell2)) + Space(80) + Substr(E3,1,40)
Endif

Case Utypep = 'T' && Title Main Entry
  Mainindex = E3 + Space(100-Len(E3)) +;
  Space(110) + Substr(E3,1,40)

Case Utypep = ' ' && No Uniform Title
  If Mtypep = 'A' && with only Author
    Mainindex = E11 + Space(100-Len(E11)) + E12 + Space(30-Len(E12));
    + Space(80) + Substr(E3,1,40)
  Endif

  If Mtypep = 'P'
    Mainindex = E11 + Space(100-Len(E11)) + E13 ;
    + Space(30-Len(E13)) + Space(80)+ Substr(E3,1,40)
  Endif

  If Mtypep = 'C'
    Mainindex = E11 + Space(100-Len(E11)) ;
    + E13 + Space(30-Len(E13)) + Space(80)+ Substr(E3,1,40)
  Endif

** No Main Entry Headings
If Mtypep = ' ',
  Mainindex = Space(210) + Substr(E3,1,40)
Endif
Endcase
Msortx = Upper(Mainindex)
RETURN
*****

Procedure Mainval
Parameters Mtypep, Mkeyx
** Uses the type of heading and the key to find matching record
Private Mopt
Mopt = 'S'

457
Select 1
Set Order to 4
Go Top
Mkeyx2 = Mtypep + Mkeyx
Seek Mkeyx2

Do while .Not. Eof().and. Mkey = Mkeyx
E1 = Space(1)
Do Ucommem && Unstrings Mem field to extract Main Entry Heading (E1)
Mainy = E1
E11 = Space(1)
E12 = Space(1)
E13 = Space(1)
E14 = Space(1)
E15 = Space(1)
E16 = Space(1)

Do Ucommmain && Unstrings heading into 6 components
E1x = Space(1)
E1x2 = Space(1)
E1x3 = Space(1)
E1x4 = Space(1)

Skelvar = 'DUPLICATE HEADINGS'

Do Case
Case Mtypep = 'A'
E1x = E11
If E16 = 'Y'.and. Len(Trim(E14)) > 0
E1x = E1x + ' ' + Trim(E14) + ' ' + Trim(E12)
E1x2 = Trim(E13)
E1x3 = Trim(E15)
Else
E1x = E1x + ' ' + Trim(E12) + ' ' + Trim(E14)
E1x2 = Trim(E13)
E1x3 = Trim(E15)
Endif

Case Mtypep = 'P'
E1x = Trim(E11)
E1x2 = Trim(E13)

Case Mtypep = 'C'
E1x = Trim(E11)
E1x2 = Trim(E12) + ' '
If Len(Trim(E13)) > 0
E1x3 = Trim(E13)
Endif
Endcase

Do Skeleton with Skelvar. 'Y', 'DUPLICATE HEADING SCREEN'
@5,0 Say 'PRESENTLY SELECTED HEADING IS: '
@6,0 Say Element1x
@7,0 Say Element2x
@8,0 Say Addsx
@9,0 Say Element3x
@10,0 Say Nobilx
@11,0 Say '******************************************************************************'
@14,0 Say 'HEADING FOUND IN SYSTEM IS :'
@15,0 Say Elx
@17,0 Say Elx2
@18,0 Say Elx3
@21,0 Say 'Accept heading, Skip Record, Quit Option (A,S,Q) : ' Get Mopt Picture !'
Read
If Mopt = 'Q'
Exit
Endif
If Mopt = 'A'
Do Case
  Case Mtypepx = 'A'
    Element1x = El1 + Space(50-Len(El1))
    Element2x = El2 + Space(50-Len(El2))
    Addsx = El3 + Space(50-Len(El3))
    Nobilx = El4 + Space(50-Len(El4))
    Britishx = El6
    @21,0 Say 'Replacing Entry Element, Forenames, Nobility and Additions'
    @22,0 Say 'and British, Relator Codes excluded, Re-edit if necessary'
  Wait
Case Mtypepx = 'P'
  Element1x = El1 + Space(100-Len(El1))
  Addsx = El3 + Space(50-Len(El3))
  @21,0 Say 'Replacing Place and Additions, Re-edit if Necessary'
  Wait
Case Mtypepx = 'C'
  Element1x = El1 + Space(100-Len(El1))
  @21,0 Say 'Replacing, Corporate Bodies and Subordinate Bodies'
  @22,0 Say 'Additions are excluded, Re-edit if necessary'
  Wait
Endcase
Exit
Endif
Skip
Enddo
RETURN
*****
Procedure Names
** initialises name variables for display. If called from Uprog
** the existing data is used and are given maximum lengths.

Skelvar = 'NAMES INPUT / UPDATE'
If Len(Trim(Element1x)) > 0 .or. Progname = 'UPROG'
E11 = Space(1)
E12 = Space(1)
E13 = Space(1)
E14 = Space(1)
E15 = Space(1)
E16 = 'N'
Mainx = Mainx
Do Uconmain && Unstrings main entry elements
   Element1x = E11 + Space(50-Len(E11))
   Element2x = E12 + Space(50-Len(E12))
   Addsx = E13 + Space(50-Len(E13))
   Noblx = E14 + Space(50-Len(E14))
   Element3x = E15 + Space(50-Len(E15))
   Britishx = E16 + Space(1-Len(E16))
Else
   Element1x = Space(50)
   Element2x = Space(50)
   Addsx = Space(50)
   Noblx = Space(50)
   Element3x = Space(50)
   Britishx = 'N'
   Mkeyx = Space(15)
   Mainx = Space(1)
Endif

Flag1 = 'T'

Do while .T.
   If Flag1 = 'N'
      wait
   Endif
Clear
Do Skeleton with Skelvar, 'Y', 'NAME SCREEN'
Flag1 = 'T'
Oopt = 'N'
@06,10 Say 'Entry Element : ' Get Element1x
@08,10 Say 'Forenames : ' Get Element2x
@10,10 Say 'Nobility : ' Get Noblx
@12,10 Say 'Additions : ' Get Addsx
@14,10 Say 'Relator Code : ' Get Element3x
@16, 10 Say 'British Term of Nobility : ' Get Britishx Picture '!
Read
If Len(Trim(Element1x)) = 0
@21,0 Say 'Entry Element is Mandatory, Quit (y/n) : ' Get Oopt Picture '!'
Read
If Qopt = 'Y'
Mtypex = ',
Element2x = Space(50)
Addx = Space(50)
Nobilx = Space(50)
Element3x = Space(50)
Exit
Else
Flag1 = 'N'
Endif
Endif

If Britishx $ 'YN' .OR. Britishx = '?'
Else
@22,0 Say 'Only two options are allowed for British Terms (Y/N)'
Flag1 = 'N'
Endif

If Britishx = '?'
@21,0 Say Space(80)
@22,0 Say Space(80)
@21,0 Say 'The field is used to order Titles of Nobility after the Entry Element'
@22,0 Say 'Use field also for Non-British Nobility requiring such ordering'
@23,0 Say 'Set field to Y, to order nobility after Entry Element'
Flag1 = 'N'
Endif

If Substr(Element1x,1,1) = '?'
@21,0 Say Space(80)
@21,0 Say 'The entry element is usually a Surname, initial or phrase'
@22,0 Say Space(80)
@22,0 Say "Enter Entry Element accordingly e.g. 'Smith' or 'X' as in Malcolm X"
@23,0 Say 'Enter articles and prepositions, if applicable in terms of AACR2R'
Flag1 = 'N'
Endif

If Substr(Element2x,1,1) = '?'
@21,0 Say Space(80)
@22,0 Say Space(80)
@21,0 Say 'Enter Initials and prefixes e.g. J, J F K, Pieter van der'
@22,0 Say 'Omit Initials if Already an Entry Element'
Flag1 = 'N'
Endif

If Substr(Nobilx,1,1) = '?'
@21,0 Say Space(80)
@22,0 Say Space(80)
@12,0 Say 'Enter Title of Nobility e.g. Sir, Dr, Lady'
461
Flag1 = 'N'
Endif

If Substr(addsx,1,1) = '?'
  @21.0 Say Space(80)
  @22.0 Say Space(80)
  @21.0 Say 'Enter Dates or Fuller forms of a name, e.g. 1945-1993'
  @22.0 Say 'e.g. (Russell Edgar) for Russell E'
  Flag1 = 'N'
Endif

If Substr(Element3x,1,1) = '?'
  @21.0 Say Space(80)
  @22.0 Say Space(80)
  @21.0 Say 'Enter the role of the author in the work e.g.'
  @22.0 Say 'Ed. and comp.'
  Flag1 = 'N'
Endif

If Flag1 = 'T'
  Exit
Endif
Enddo

Stffin = Element1x
Do Stff with Stffin & & Removes punctuations from Element1x
Stffin = Stffin + Space(15)
Stffin = Rtrim(Substr(Stffin, 1,12))
Sarr1 = Space(10)
Sarr2 = Space(10)
Keyx = Space(6)
Elx = Element2x
Do wcomp2 with Elx, Keyx & & Compresses initials that form part of key
& & Using variables SARR1 and SARR2

Mkeyx = Ltrim(Trim(Stffin) + : * + Substr(Sarr1,1,1) + Substr(Sarr2,1,1))

If Progname = 'IPROG' .and. Len(Trim(Mkeyx)) > 1
Do Mainval with Mtypex, Mkeyx & & Names validated against those in
& & system using Mainval
Mainx = Space(1)
Mainy = Space(1)

** Main entry heading composed for storage purpose
Do Connmain with Element1x, Element2x, Addsx, Mobilx, Element3x, Britishx
Mainx = Mainy
RETURN

*****
Procedure Notes
** Input and Updates Notes
** Using Sx and Sy as variables

P2 = 65
Skelvar = 'NOTES INPUT/UPDATE MENU'

Do While P2 > 64 .and. P2 <99
   Ax = Ltrim(Str(P2))
   Sx&Ax = Space(1)       & Initialising variables
   P2 = P2 + 1
Enddo

If Progname = 'IPROG'
   Else
      Do Uconnote
      Endif

Do While .T.
   Do While .T.
      Clear
      Noteopt = 'Z'
      @ 4,0 say ""

      TEXT
      A In Analytic
      B Relationship Complexity
      C Nature, Scope, form
      D Language
      E Source of Title Proper
      F Variation in Title
      G Parallel Title and
      H Other Title Information
      I Statement of responsibility
      J Donor, source, previous Owners
      K Published Versions
      L Physical Description
      M Accompanying Material
      N Dissertations/Thesis
      O Access and Literary Rights
      P Reference to Published Descriptions
      Q Summary
      R Contents
      S Ancient, medieval and
      T Renaissance Ms.
      U Biographical/Historical
      V Placement with other Collections
      W Miscellaneous
      X Miscellaneous (NAREM)

   Endtext

   Do Skeleton with Skelvar, 'N', 'NOTES SCREEN'
   @19,0 Say 'Select Note to Add or Update (Z to Exit)' Get Noteopt Picture '!''
   @19,45 Say 'To Browse Notes Enter ?'
   @21,0 Say 'To view ABSTRACT for NAREM and AACR2R select 0 and 1 respectively'
   READ
   If Noteopt $ 'ABCDEFGHIJKLMNOPQRSTUVWXYZ01'
       Exit
   Endif
Enddo

463
If Noteopt = 'Z' .or. Noteopt = '0' .or. Noteopt = '1'
Exit
Endif

If Noteopt = '?'
Do Brownnote
Endif
Ax2 = Asc(Noteopt) && Converts selected option into Asc character
Ax = Ltrim(Str(Ax2)) && which in turn is converted to a string Ax
Do Skeleton with Skelvar, 'y', 'NOTES SCREEN'

If Ax2 >64
If Len(Sx&Ax) < 2
Sx&Ax = Space(254)
Anynote = Space(254)
Endif
If Clipper
if Ax = '81'
Anynote = Sx&Ax
Do Skeleton with 'A B S T R A C T - N O T E S', 'Y', 'ABSTRACT'
@21,0 Say Space(80)
@22,0 Say Space(80)
@21,0 Say 'Press ENTER after each new line is entered while in INSERT mode'
@22,0 Say 'A full screen edit is available, use ESC key to EXIT WITHOUT SAVING'
@23,0 Say 'To SAVE press Control + W after abstract is entered'
Anynote = memoedit(Anynote,5,12,19,78,.T.)
else
Anynote = Sx&Ax + Space(800 - Len(Sx&Ax))
Do Skeleton with Skelvar, 'Y', 'NOTES SCREEN'
@5,0 Say 'Enter Note : ' Get Anynote
Read
Endif
Endif

Else
Anynote = Sx&Ax + Space(254-Len(Sx&Ax))
@12,10 Say 'Enter Note : ' Get Anynote
Read
Endif
Sx&Ax = Trim(Anynote)
Endif
Enddo
Do Connote && Concatenates notes as one string for storage
If Noteopt = '0'
Ax = '81'
Abstract= Substr(Sx&Ax,2,Len(Sx&Ax)-1) && View format for Narem abstract
Clear
? 'ABSTRACT :
? Abstract
Wait
Endif

464
If Noteopt = '1'
  Do Summary & View format of ACR2R Summary
  Endif
  RETURN

*****

Procedure Place
** Inputs and Updates Places

  Skelvar = 'PLACE NAMES INPUT/UPDATE'
  If Len(Trim(Elementlx)) > 0 .or. Progname = 'UPROG'
    E11 = Space(1)
    E13 = Space(1)
    Mainy = Mainx
    Do Uconnamx & Fills fields E1 and E3
      Elementlx = E11 + Space(100-Len(E11))
      Addsx = E13 + Space(50-Len(E13))
    Else
      Elementlx = Space(100)
      Addsx = Space(50)
    Endif

    Flag1 = 'T'
  Do While .T.
    If Flag1 = 'N'
      wait
    Endif

    Do Skeleton with Skelvar, 'Y', 'PLACE SCREEN'
      Qopt = 'N'
      Flag1 = 'T'
      @10,10 Say 'Place : ' Get Elementlx
      @14,10 Say 'Additions : ' Get Addsx
      Read

      If Len(Trim(Elementlx)) = 0
        @22,0 Say 'A Place Name (e.g. France) is Mandatory, Quit (Y/N) : ' Get Qopt
        Picture !'
        Read
        If Qopt = 'Y'
          Elementlx = Space(100)
          Addsx = Space(50)
          Wtypep = ','
        Exit
        Else
          Flag1 = 'N'
        Endif
      Endif

      If Qopt = 'N'
        Elementlx = Elementlx + Space(100)
If Substr(Element1x,1,1) = '?
@22,0 Say Space(80)
@22,0 Say 'Enter a place e.g. France'
Flag1 = 'N'
Endif

If Substr(Addsx,1,1) = '?
@21,0 say Space(80)
@22,0 Say Space(80)
@21,0 Say 'Enter additions, e.g. Hungary as in Budapest (Hungary)'
Flag1 = 'N'
Endif

If Flag1 = 'T'
Exit
Endif
Enddo

Keyx = Space(6)
Elx = Elementlx
Do wcomp2 with Elx, keyx  && Removes punctuations and
Mkeyx = Keyx          && constructs key for Place
If Progname = 'IPROG'.and. Len(Trim(Mkeyx)) > 1
Do Mainval with Mtypex, Mkeyx  && Validates Place Names
Endif           && against those in database

Mainx = Space(1)
Mainy = Space(1)
** Constructs Main Entry Heading Storage Format
Do Connmain with Elementlx, "", Addsx, "", "", ""
Mainx = - Mainy
RETURN

******

Procedure Screen3
** Procedure Inputs and Updates the edition, dates,
** Physical description, Reference, classification
** NAREM status, type and depot code

Ref1x = Space(10)
Ref2x = Space(10)
Skelvar = 'UTILITY SCREEN - INPUT/UPDATE'

If Progname = 'UPROG'
Depotx = Depotx + Space(20-Len(Depotx))
Edx = Edx + Space(50-Len(Edx))
Datx = Datx + Space(40-Len(Datx))
Date1x = Date1x + Space(8-Len(Date1x))
Date2x = Date2x + Space(8-Len(Date2x))
Phdescx = Phdescx + Space(60-Len(Phdescx))
Refx = Refx+ Space(20-Len(Refx))
Ref1x = Substr(Refx,1,10)
Ref2x = Substr(refx,11,10)
Classx = Classx + Space(20-Len(Classx))
Endif
DcodeX = Depotx

Dopt = 'O'
Tflag = .T.

Do While .T.
   If Tflag
      Else
         wait
      Endif

Clear
Do Skeleton with skelvar, 'Y', 'UTILITY SCREEN'
Tflag = .I.
@5,0 Say 'Edition : ' Get Edx
@7,7 Say 'Dates (FrGe Format) : ' Get Datx
@9,0 Say 'Starting Date (mm/dd/year) : ' Get Date1x Picture '999999999'
@9,42 Say 'Ending Date (mm/dd/year) : ' Get Date2x Picture '999999999'
@11,0 Say 'Physical : ' Get Phdescx
@12,0 Say 'Description : ' Get Phdscx
@13,0 Say 'Reference Number : ' Get Ref1x Picture '!!!!!!!!!!!'
@13,30 Get Ref2x Picture '999999999'
@14,13 Say '[Alpha portion]'
@14,29 Say '[Numeric portion]'
@13,44 Say 'Depot Code : ' Get DcodeX
@16,0 Say 'NAREM Entry(Y/N) : ' Get Naremx
@16,40 Say 'Classification : ' Get Classx Picture '!!!!!!!!!!!!!!!!!!!!!!'
@18,0 Say 'Type of Document (Manuscript or Accessions [M/A]) : ' Get Typex
Picture '
Read
Refx = Ref1x + Space(10-Len(Ref1x)) + Ltrim(Ref2x)

If Len(Trim(Datx)) = 0
   @19,0 Say 'A Free Format Date must exist between Title and Date area (Override) : ' Get Dopt Picture '/'
   Read
   @21,0 Say Space(80)
   @22,0 Say Space(80)
   If Dopt <> 'O'
      Tflag = .F.
   Endif
   Endif

** Help Options for edition, dates, physical description, reference
If Substr(Edx,1,1) = '?' && Help on edition
@22,0 Say Space(80)
@22,0 Say "Enter edition if one exists e.g. '2nd rev. ed.'"
Tflag = .N.

467
Endif
If Substr(Date1x,1,1) = ' ' &amp; Help on dates
@23,0 Say Space(80)
@23,0 Say "Enter a free format date e.g. '21 May 1983'"
Tflag = .N.
Endif

If Substr(Phdescx,1,1) = ' ?'
@23,0 Say Space(80)
@23,0 Say "Enter physical description e.g. '25 boxes'"
Tflag = .N.
Endif

If Substr(Ref1x,1,1) = ' ?'
@22,0 Say Space(80)
@22,0 Say 'Enter up to 10 alphabetic characters for REFERENCE numbers'
@23,0 Say Space(80)
@23,0 Say 'e.g. a, AB etc. followed by serial numbering e.g. 1,2, 3, etc.'
Tflag = .N.
Endif

** Testing Dates
Dflag = .T.
Mm = Substr(Date1x,1,2)
Dd = Substr(Date1x,3,2)
Yr = Substr(Date1x,5,4)

If val(Mm) &gt; 12 &amp;&amp; Invalid month
Dflag = .N.
Endif
If val(Dd) &gt; 31 &amp;&amp; Invalid day
Dflag = .N.
Endif
If Mm = '00'
Mm = '
Endif
If dd = '00'
dd = '
Endif

** Testing if day or month &gt; 0 while year zero - date declared invalid
If yr = '0000'
If Len(Trim(Mm)) &gt; 0 .or. Len(Trim(dd)) &gt; 0
Dflag = .N.
Endif
Endif

** Testing second set of dates
Mm = Substr(Date2x,1,2)
Dd = Substr(Date2x,3,2)
Yr = Substr(Date2x,5,4)

If val(Mm) > 12    && Invalid month
Dflag = .N.
Endif

If val(Dd) > 31    && Invalid Day
Dflag = .N.
Endif

If mm = '00'
mm = ',
Endif
If dd = '00'
dd = ',
Endif

** Testing if day and month > 0 while year zero - date declared invalid
If yr = '0000'
If Len(Trim(mm)) > 0 .or. Len(Trim(dd)) > 0
Dflag = .N.
Endif
Endif

If Dflag
Else
@21,0 Say 'Invalid Dates Entered, Retry'
Tflag = .F.
Endif

** Testing Depot Code
Depflag = .T.
If Substr(Dcodez,1,1) = '?'
Do Depot
Else
Do Depotval
Endif

If Depflag
Else
@21,0 Say Space(80)
@21,0 Say 'Invalid Depot Code Entered, To Query Type ? for code'
Tflag = .F.
Endif

If Typex = 'M'.or. Typex = 'A'
Else
Tflag = .F.
@22,0 Say Space(80)
@22,0 Say 'Valid Type of Document are M or A'
Endif

469
If Type$ = 'M' && Help on Type
Tflag = .F.
022.0 Say Space(80)
022.0 Say "Only two types of documents are allowed, 'M' for Manuscripts"
023.0 Say "and 'A' for Accessions (archives)"
Endif

If Tflag && When no errors detected loop exited
Exit
Endif
Enddo

Depotx = Dcodex
RETURN

*****

Procedure Skeleton
Parameters Skelvar, Skelvar2, SKELVAR3
** Procedure sets up screen layout

If Skelvar2 = upper('y') .or. Skelvar2 = 'y'
Clear
Endif
SKL = Replicate ('*', 79)
0.0 Say SKELVAR3
01.0 Say SKL
02.0 Say '*'
02.78 Say "*
03.0 Say Sk1
Do Centre With 2, Skelvar && Centres the heading
0.65 Say Date()
020.0 Say Sk1
RETURN

*****

Procedure Stff
Parameters Stffin
** Procedure Stuffs string D_8 removes all punctuations
D_8 = Trim(Upper(Stffin))

** Removes punctuations and odd characters when generating key
P_1 = 1
Do While P_1 <= Len(D_8) .And. Len(Trim(D_8)) > 0
  S_1 = SubStr(D_8, P_1, 1)
  If Asc(S_1) < 48 .OR. Asc(S_1) > 90 .OR. (Asc(S_1) > 57 .And. Asc(S_1) < 65)
    D_8 = Stuff(D_8, P_1, 1,"")
  Else
    P_1 = P_1 + 1
Enddo

470
Endif
Enddo
Stffin = D_B
RETURN

*****

Procedure Summary
**Aim Views format of AACR2R Summary
ax = '81'
yyy = Substr(Sx&ax,2,Len(Sx&Ax)-1) + Chr(13) + Chr(10)
xx = 1
mline = 1
Clear
? 'Summary : '
?----------
? do while Len(yyy) > 0
Mchars = Substr(yyy,1,At(Chr(13)+Chr(10),yyy))
If Substr(Mchars,1,1) = '+'
mchars = Substr(Mchars,2,Len(Mchars)-1)
Endif
If mline > 1
? Rtrim(Ltrim(Mchars))
Else
? Space(4) + Rtrim(Ltrim(Mchars))
Endif
yyy = Substr(yyy, At(chr(13),yyy)+2, Len(yyy)- At(chr(13),yyy))
xx = xx +1
If xx > 15
wait
clear
xx = 1
Endif
Mline = mline + 1
Enddo
Wait
RETURN

*****

Procedure Uconmain
** Unstrings subfields of Main Entry Heading using variable Mainy
** Fields E11-E16 filled into appropriate variables e.g entry element
** Complements Procedure conmain.

Loopcnt = 7
Private D_B

471
Private P.1
Private P.2
Private P.2X
Private H.1
Lword = 'F'
P.1 = 1
P.2 = 1
H.1 = "..
E11 = space(1)
E12 = Space(1)
E13 = Space(1)
E14 = Space(1)
E15 = Space(1)
E16 = Space(1)

** Now removing first delimiter, Chr(181) from Mainy
If Len(Mainy) > 1 .and. Substr(Mainy,1,1) = Chr(181)
    Mainy = Substr(Mainy,2,Len(Mainy)-1)
Endif

** Each field (E11-E16) is computed and isolated
Do While P.2 <= Loopcnt .And. Len(Trim(Mainy)) > 0
    Mainy = Ltrim(Rtrim(Mainy))
    If Chr(181) $Mainy
        D.8 = Substr(Mainy,1,At(chr(181), Mainy)-1)
        If Len(D.8) + 1 = Len(Mainy)
            Lword = 'T'
        Else
            Mainy = Substr(Mainy, At(chr(181), Mainy) + 1, Len(Mainy) - AT(chr(181), ;
            Mainy))
    Endif
    Else
        Lword = 'T'
        D.8 = Substr(Mainy,1,Len(Mainy)-1)
        If Len(Mainy) <2
            Exit
        Endif
    Endif

Bx = Ltrim(Str(P.2))

If P.2 >7
    Clear
    @12.12 Say 'Irrecoverable Error on Record Structure, Aborted'
    Wait
    Clear
Endif

** Depending on the value of Bx, the elements E11-E16 are filled
If P.2 >0 .And. P.2 < 7
    Do Case
        Case Bx = '1'

472
E11 = D_8
Case Bx = '2'
E12 = D_8
Case Bx = '3'
E13 = D_8
Case Bx = '4'
E14 = D_8
Case Bx = '5'
E15 = D_8
Case Bx = '6'
E16 = D_8
Endcase
P_2 = P_2 + 1
Endif

If Lword = 'T'
Exit
Endif
Enddo
RETURN

*****

Procedure Uconmem
** Procedure Unstrings mem field using variable conmemx
** The Main, Uniform, Title, Edition, Physical Description
** Date and Notes are filled into variables E1-E7.

If Clipper
Conmemx = Mem
Else
Conmemx = Space(1)
Endif
Loopcnt = 8
Private D_8
Private P_1
Private P_2
Private P_2X
Private H_1
Lword = 'F'
P_1 = 1
P_2 = 1
H_1 = "-"
E1= Space(1)
E2 = Space(1)
E3 = Space(1)
E4 = Space(1)
E5 = Space(1)
E6 = Space(1)
E7 = Space(1)

** Now removing first delimiter, Chr(180) from Conmemx
473
If Len(Commemx) > 1 .and. Substr(Commemx,1,1) = Chr(180)
Commemx = Substr(Commemx,2,Len(Commemx)-1)
Endif

** Each field (E1-E7) is computed and isolated
Do While P_2 <= Loopcnt .And. Len(Trim(Commemx)) > 0
Commemx = Ltrim(Rtrim(Commemx))

If Chr(180) $Commemx
    D_8 = Substr(Commemx,1,At(Chr(180), Commemx)-1)
    If Len(D_8) + 1 = Len(Commemx)
        Lword = 'T'   && Determines if last word being processed
    Else
        Commemx = Substr(Commemx, At(Chr(180), Commemx) + 1, Len(Commemx) -
                        At(Chr(180), Commemx))
    Endif
Else
    Lword = 'T'
    D_8 = Substr(Commemx,1,Len(Commemx)-1)
    If Len(Commemx) < 2   && Exits when length = 1 (delimiter)
        Exit
    Endif
Endif

Bx = Ltrim(Str(P_2))

If P_2 > 7
    Clear
@12.12 Say 'Irrecoverable Error on Record Structure, Aborted'
    Wait
    Clear
Endif

** Filling each field depending on value of Bx
If P_2 > 0 .and. P_2 < 8
    Do Case
        Case Bx = '1'
            E1 = D_8         && Main Entry Heading
        Case Bx = '2'
            E2 = D_8         && Uniform Title
        Case Bx = '3'
            E3 = D_8         && Title
        Case Bx = '4'
            E4 = D_8         && Edition
        Case Bx = '5'
            E5 = D_8         && Physical Description
        Case Bx = '6'
            E6 = D_8         && Dates
        Case Bx = '7'
            E7 = D_8         && Notes
    Endcase
    P_2 = P_2 + 1
Endif

474
If Lword = 'T'
   Exit && Exits when last subfield processed
Endif
Enddo
RETURN

*****

Procedure Uconnote
** Unstrings Connotex into respective Notes, indicators ranging from A to X.
Anyflag = .F. && Used to keep track of errors on indicators
Loopcnt = 30
Private D_8
Private P_1
Private P_2
Private P_2X
Private H_1
Lword = 'F'
P_1 = 1
P_2 = 1
H_1 = "_."

*** Initialises Variables To Hold Notes
P2 = 65

Do While P2 > 64 .and. P2 <89
Ax = Ltrim(Str(P2))
Sx&Ax = Space(1)
P2 = P2 + 1
Enddo

** Now Filling Each Note Using Chr(181) as Delimiter
** Leading delimeter removed in second line
If Len(Connotex) > 1 .and. Substr(Connotex,1,1) = chr(181)
Connotex = Substr(Connotex,2,Len(Connotex)-1)
Endif

** Tests for existence of Chr(181) and fills D_8 up to next delimiter
Do While P_2 <= Loopcnt .And. Len(Trim(Connotex)) > 0
Connotex = Ltrim(Rtrim(Connotex))
If Chr(181) $Connotex
   D_8 = Substr(Connotex,1,At(Chr(181), Connotex)-1)
   If Len(D_8) + 1 = Len(Connotex)
      Lword = 'T' && Tests if last word
   Else
      Connotex = Substr(Connotex, At(Chr(181), Connotex) + 1, Len(Connotex))
   Endif
Else
   Lword = 'T'
   D_8 = Substr(Connotex,1,Len(Connotex)-1)
   If Len(Connotex) <2 && Field must be at least 2 characters

475
& & One for delimiter other for note.

Endif

P2X = Substr(D_8,1,1) & & Indicator is extracted
Ax = Ltrim(Str(Asc(P2x)))

If Asc(P2x) > 64 . and. Asc(P2x) < 89
    Sx&Ax = Substr(D_8,2,Len(D_8)-1) & & Variable filled with contents of D_8
    & & Indicator excluded.
Else
    If Anyflag
        Clear
        @12,15 Say 'Too Many errors Generated from incorrect Indicators, Process Aborted'
        Wait
    Exit
Endif
Ax = '87'
Sx&Ax = Substr(D_8,2,Len(D_8)-1) & & Assigned miscellaneous category
        & & when valid indicator not found
        & & Variable with indicator 'W' filled
Endif

P_2 = P_2 + 1
If Lword = 'T'
    Exit & & Forces exit when last note processed
Endif
Enddo
RETURN

*****

Procedure Uniform
** Inputs and Updates Uniform Titles

Skelvar = 'UNIFORM T I T L E S : I N P U T / U P D A T E'
If Len(Trim(Elementl1x)) > 0 . OR. Progname = 'UPROG'
    If Len(Trim(Uniformx)) > 3
        Upunct = Substr(Uniformx,Len(Uniformx)-1,1)
        If Upunct $','.(
            Else
            Upunct = '('
Endif
Uniformx = Substr(Uniformx,2,Len(Uniformx)-1)
Elementl1x = Substr(Uniformx,1, At(Chr(181), Uniformx)-1)
Elementl1x = Elementl1x + Space(100- Len(Elementl1x))
Addsl1x = Substr(Uniformx, At(Chr(181), Uniformx)+1, Len(Uniformx)-
        At(Chr(181),Uniformx)-3)
        Addsl1x = Addsl1x + Space(50- Len(Addsl1x))
        Else
        Elementl1x = Space(100)

476
Addslx = Space(50)
Upunct = '('
Endif
Else
Element1lx = Space(100)
Addslx = Space(50)
Upunct = '('/
Endif
Utypeex = 'U'
Ukeyx = Space(6)
Flagl = 'T'
Do While .T.
   If Flagl = 'N'
      wait
   Endif
   Do Skeleton with Skelvar, 'Y', 'UNIFORM SCREEN'
      Flagl = 'T'
      Qopt = 'N'
      @4,5 Say 'Uniform Title : ' Get Element1lx
      @7,5 Say 'Additions : ' Get Addslx
      @8,5 Say 'Punctuation Separator : Acceptable values are ".(,": Get Upunct
      @9,10 Say '---------------------------------------------------------------'
      @11,0 Say 'Enter the Uniform Title ONLY (omit Places, Names or Corporate'
      @12,0 Say 'Bodies which are entered SEPARATELY under the Place, Names'
      @13,0 Say 'or Corporate Body - See previous screen.'
      @14,0 say 'Uniform Titles must NOT be in Square Brackets'
      @15,0 Say 'as this affects filing,'
      @18,0 Say 'Example 1 : Constitution '
      @19,0 Say 'Example 2 : Bible. O.T. Genesis XII'
   Read
   If Len(Trim(Element1lx)) = 0
      @22,0 Say 'Entry Element is Mandatory, Quit (Y/N) : ' Get Qopt Picture '!
   Read
      If Qopt = 'Y'
         Element1lx = Space(1)
         Addslx = Space(1)
         Utype = ' '
      Exit
   Else
      Flagl = 'N'
   Endif
Endif
If Upunct $'.,'( '
Else
   @22,0 Say Space(80)
   @22,0 Say 'Valid Punctuations are ., and ('
Flag1 = 'N'
Endif

If Upunct $'?'
@23,0 Say Space(80)
@23,0 Say 'Valid Punctuations are , and ( see AACR2R'
Flag1 = 'N'
If Upunct = '?'
Upunct = ',',
Endif
Endif

If Substr(Element11x,1,1) = '?'
@21,0 Say Space(80)
@22,0 Say Space(80)
@21,0 Say 'Enter Uniform Title e.g. Bible. O.T.'
@22,0 Say 'Enter Name, Place or Corporate Body under separate Options provided'
Flag1 = 'N'
Endif

If Substr(Addsl1x,1,1) = '?'
@21,0 Say Space(80)
@22,0 Say Space(80)
@21,0 Say "Example of Additions 'English' as in 'Teorema. English'
@22,0 Say "Example : Parts : 'Book 1-6' as in 'Iład. Book 1-6'''
Flag1 = 'N'
Endif

If Flag1 = 'T'
Exit
Endif
Enddo

Keyx = Space(6)
Elx = Element11x
Do Wcomp2 with Elx, Keyx && Composes key for Uniform Title
Ukeyx = Keyx

Ukeyx = Upper(Ukeyx)
If Progname = 'IPROG'.and. Utypep = 'U'.and. Len(trim(Ukeyx)) > 0
Do Unval with Ukeyx && Validates Uniform Titles
Endif

** Constructs Uniform Heading for storage
Uniformx = Chr(181) + Trim(Element11x) + Chr(181) + Trim(Addsl1x);
+ Chr(181) +Trim(Upunct) + Chr(181)
RETURN

*******

478
Procedure Unval
Parameters Ukeyx
** Procedure validates Uniform titles
Private Mopt
Mopt = 'S'
Select 1
Set Order to 5
Go Top
Ukeyx2 = Utypex + Ukeyx
Seek Ukeyx2
Clear
Do While .Not. EOF() .and. Ukey = Ukeyx
E2 = Space(1)
Do Ucommem

E2 = Trim(E2)
E2 = E2 + Space(100-Len(E2))
E11 = Substr(E2,2,Len(E2)-1)
E11 = E11 + Space(100-Len(E11))
E12 = Substr(E2,At(Chr(181),E2)+1,Len(E2)-At(Chr(181),E2)-2)
E12 = Ltrim(Trim(E12))
E12 = Substr(E12,1,Len(E12)-3) & & Excluding 2 delimiter and Punctuation
E12 = E12 + Space(50-Len(E12))

Skelvar = 'D U P L I C A T E H E A D I N G S'
Do Skeleton with Skelvar, 'Y', 'DUPLICATE HEADING SCREEN'
@6.0 Say 'PRESENTLY SELECTED HEADING IS : '
@7.0 Say Element11x
@9.0 Say Addsx
@11.0 Say '********************************************************************'
@13.0 Say 'HEADING FOUND IN SYSTEM IS : '
@14.0 Say E11
@16.0 Say E12
@18.0 Say 'Accept heading, Skip Record, Quit Option (A,S,Q) : ' Get Mopt Picture '/!'
Read
If Mopt = 'Q'
Exit
Endif

If Mopt = 'A'
Element11x = E11
Addsx = E12
@21.0 Say 'Replacing Entry Element and Additions'
@22.0 Say 'Re-edit if necessary'
Wait
Exit
Endif
Skip
Enddo
RETURN
****

Procedure WCOMP2
** Procedure unstrings text into words using blank as delimiters.
** Punctuations are removed. Up to 6 words are isolated.
** PARAMETERS D_1, keyx
  Loopcnt = 12
  Private D_8
  Private P_1
  Private P_2
  Private P_2X
  Private H_1
  Lword = 'F'
  D_1 = Upper(D_1)
  P_1 = 1
  P_2 = 1
  Wdcnt = 0
  H_1 = ""

  Sarr1 = Space(10)
  Sarr2 = Space(10)
  Sarr3 = Space(10)
  Sarr4 = Space(10)
  Sarr5 = Space(10)
  Sarr6 = Space(10)

  *** isolates a word using spaces as delimiter
  Do While P_2 <= Loopcnt .And. Len(Trim(D_1)) > 0
    D_1 = Ltrim(Rtrim(D_1))
    If ' ' =D_1
      D_8 = Substr(D_1,1,At(" ", D_1))
      D_1 = Substr(D_1, At(" ", D_1) + 1, Len(D_1))
    Else
      Lword = 'T' & & This is the last word
      D_8 = D_1
      If Len(D_1) = 0
      Exit
    Endif
  Enddo

  ** Removes punctuations and odd characters in word, D_8
  P_1 = 1
  Do While P_1 <= Len(D_8) .And. Len(Trim(D_8)) > 0
    S_1 = Substr(D_8, P_1, 1)
    If Asc(S_1) < 48 .OR. Asc(S_1) > 90 .OR. (Asc(S_1) > 57 .and. Asc(S_1) < 65)
      D_8 = Stuff(D_8, P_1,1,"")
    Else
      P_1 = P_1 + 1
    Endif
  Enddo

  *** Using 'Sarr?' variables to store the words

  480
If Len(Trim(D8)) > 0
Wdcnt = Wdcnt + 1
D8 = D8 + Space(20)
D8 = Substr(D8,1,10)
P2X = Ltrim(Str(P2))
Sarr&P2X = D8
P_2 = P_2 + 1
Endif

If Lword = 'T'
Exit
Endif
Enddo

** Now forming key from first three characters of first word
** One character each from 2nd, 3rd and 4th words.
Keyx = Rtrim(Substr(Sarr1,1,3))
If Len(Rtrim(Sarr2))>0
Keyx = keyx + Rtrim(Substr(Sarr2,1,1))
Endif
If Len(Rtrim(Sarr3))>0
Keyx = keyx + Rtrim(Substr(Sarr3,1,1))
Endif
If Len(Rtrim(Sarr4))>0
Keyx = keyx + Rtrim(Substr(Sarr4,1,1))
Endif
RETURN

******
REFERENCE

2nd rev. ed. Canadian Library Association  
: Ottawa.


487

Cook, M. *The management of information from archives.* Aldershot: Gower.


489


*S.A Archives Journal*, 33: 34-46.


492


Journal of Librarianship, 10(2) : 131-137.


Hopkinson, Alan. 1990. Information transfer and exchange formats in McIlwaine, I C ed. *Standards for the international exchange of bibliographic information, papers presented at a course held*


497
Killie Campbell Africana Library. 1989. *Basic
guideline for manuscript collection
JFD/KG).

introduction to systems analysis and design.*
2nd ed. New Jersey: Prentice-Hall.

Knapp, J.F. 1968. Design considerations for the
Marc magnetic tape formats. *Library Resources
and Technical Services, 12*(3) : 275-85.

Koenig, M.E.D. 1985. Data relationships: bibliographic information retrieval systems and
database management systems. *Information

Layne, S.S. 1989. Integration and the objectives of
the catalog. In Svenonius, E ed. *The conceptual
foundations of Descriptive Cataloging*, pp. 185-


Vervliet, H.D.L. 1976. ISDS, ISBD(S), and ISBD(M) as cataloging aids. *Journal of Library Automation*, 9(2) : 137-144.


507