Commercial microcomputer database management system software evaluation guidelines

by

Nicholas Kassem

Information Sciences Division
International Development Research Centre
Ottawa, Canada

Prepared for the

meeting on

Microcomputers and Bibliographic Information Systems in Latin America: Problems, Experiences and Projections

U.N. Economic Commission for Latin America
Santiago, Chile

April 1984

Commercial microcomputer database management system software evaluation guidelines

These guidelines were written by Nicholas Kassem and were discussed by a technical working group which met in Ottawa during the period 28 November to 2 December 1983. The group was composed of analysts from IDRC, IICA, CEPIS, and BIREME, as well as a independent consultant. The current version of this report incorporates their recommendations and suggestions.

ABSTRACT

A series of guidelines to be used in the evaluation of commercial microcomputer database management system software for bibliographic applications is proposed. A high-level functional specification for a "typical" bibliographic application is given as a focus for the evaluation. The features of software to be considered in the evaluation process are defined in both general and specific terms. They include: the user interface, database definition, database population, data modification, data retrieval, arithmetic computation, output generation, data integrity, utilities and special features, the software and hardware environment, and data transfer. Particular characteristics of these functions which are useful for evaluation purposes are identified. The emphasis throughout is on the discussion of possible discriminating factors, rather than on mechanistic procedures to be followed in carrying out an actual evaluation. However, two sets of forms for "paper" evaluations are presented. One of these contains the results of an evaluation of a well-known database management package as it could be applied to a "DEVSIS-like" system.

CONTENTS

1. INTRODUCTION

There are, as yet, very few commercially available microcomputer software packages dedicated to managing primarily bibliographic information. In addition, the commercial microcomputer software producers are unlikely to address, in the short term, the specific requirements and needs of bibliographic systems. This is primarily due to the relatively small potential user base and, therefore, market for such products. The needs of these users are more likely to be satisfied through general purpose software packages which can be specially tailored to their specific requirements. Although this approach undoubtedly represents a compromise, it is nevertheless a realistic and cost effective way of balancing needs against solutions.

Bibliographic data belong to the general class of non-numeric or textual data and, therefore, for the purposes of this study the term "bibliographic data" is used in the broad sense. The software products presently available to manage such data emphasize, for obvious reasons, their capabilities rather than their limitations. In many cases, however, it is these very limitations that are of paramount importance in the selection of software for bibliographic applications. Appendix A contains the high-level functional specifications of a typical bibliographic application.

The evaluation of available software or a comparative study of selected software is beyond the scope of this document. The intention is, therefore, to present guidelines on how to evaluate microcomputer software and what specifically to look for.

1.1 Terminology

In the field of microcomputer software there is a great deal of confusion over the terms database, DBMS, FMS, NDBMS, HDBMS, etc. Some of this confusion is strictly the result of misusing terminology, which has resulted in vendors claiming features and characteristics, which is misleading to say the least.

The approach adopted has been firstly, to define the terminology and secondly, to use the terminology consistently. When evaluating any given software, the preliminary step must always involve identifying the terminology and understanding how it is being used. The following definitions and descriptions will be of assistance.

(i) Database

A database is essentially a collection of interrelated data with differing record types, stored to control redundancy and designed to serve one or more applications.

(ii) Database Management System (DBMS)

A DBMS can be thought of as a collection of software for managing one or more databases, and should provide the necessary tools to build and support mutually independent applications.

(iii) File

A file is composed of a sequence of records of predefined format. A file is uniquely identified by a name which is an entry in the system directory, and it is this name that is used to locate data on disk. A file can contain various types of data, i.e. pointers, indexes, bibliographic data and so on. It is, therefore, apparent that a database represents a logical view of specific types of data whilst a file represents a physical means of referring to (and hence accessing) a variety of data types.

(iv) Record

A record represents a collection of logically related fields grouped together into a unit. A record is, therefore, the smallest retrievable unit within a file. (In the relational model, the term tuple is used instead of record.)

(v) Field

A field represents a location within a record which has associated with it a set of attributes, e.g. length, data type, descripton, and so on. The terms domain, data element, and field are at times used interchangeably, and though each has a specific meaning and usage, I do not wish to confuse the issue by introducing such subtleties.

1.2 Methodology

In order to be able to evaluate database software packages for the purposes of managing bibliographic data, one must look at the software from various perspectives. The usefulness and, therefore, suitability of any given software product in addressing the needs of bibliographic applications is directly related to the way in which the product performs a set of previously identified functions.

The proposed methodology for evaluating micro DBMS software is made up of the following steps:-

1. Identification of the set of functions associated with a specific application;

2. Ranking of the identified functions based on whether the functions are required, desirable or optional;

3. Identification of the software features that are likely to perform the selected functions either directly or indirectly;

4. Identification of the functions that cannot be performed by the software;

5. Determination of the way in which the software features either directly or indirectly perform the selected functions;

6. Identification of the trade-offs and compromises needed to perform the set of functions using the software features provided;

7. Determination of the suitability of the product based on the results of 3), 4), 5) and 6).

Steps 1) and 2) are relatively easy to perform and the system functional specifications should be used as the starting point. In order to perform steps 3) and 4), software features and their characteristics need to be considered in detail, and to this end sections 2 - 12 of this report can be used as a guideline.

The features have been divided into the following broad categories:-

- User interface
- Database definition
- Database population
- Data modification
- Data retrieval
- Arithmetic computation
- Output generation
- Data integrity
- Utilities and special features
- Software and hardware environment
- Data transfer.

These categories are discussed in the following sections and the amount of detail has been maintained at a level to ensure usefulness without being specific to the point of detracting from the overall objectives of this document. In addition, the features serve as a checklist against the set of requirements that need to be satisfied.

Note: It should be emphasized that, although many DBMS software provide a fixed number of features, in addition they generally provide facilities, e.g. ultra languages and/or programming language interfaces, that enable a great many functions to be performed indirectly.

1.3 Database types

Detailed descriptions of various database types and internal structures are well documented and widely available. However, in order to assess various database software packages, with any degree of confidence in the validity of the exercise, it is essential to appreciate the fundamental differences between types of databases. These differences have a major bearing on the capabilities and features offered by database software packages.

The databases that are now available on microcomputers are often subsets of databases that are well established on mainframe and minicomputers. There are three broad categories: File Management Systems (FMS), Hierarchical/Network Database Management Systems (H/NDBMS) and Relational Database Management Systems (RDBMS). Refer to Appendix B for a description of the main types of commercial DBMS software.

2. USER INTERFACE

The features of a DBMS that are collectively known as the "user interface" are of an ever-increasing importance. The reason is partly due to the continuous rise in user expectations, but more importantly it is the user interface that ultimately determines the usefulness of the product and its acceptability by the user community. It is, therefore, not surprising that a great deal of effort is being put into this aspect of the total software package, the DBMS software being only a part of the whole system.

Voice recognition/synthesis, touch-sensitive screens, and optical pens are likely to be common features integrated into DBMS software in the not-too-distant future. The desirability of a natural language interface, with a limited but expandable vocabulary, is an obvious attraction. I will, however, only address the more conventional features that are currently available.

2.1 Menu driven

The ability to select functions by choosing options from a menu has become an increasingly popular method of performing various DBMS tasks. The displaying of an entire menu each time a functional decision has to be made can be cumbersome in an environment where the computer dialogue is via a hardcopy device rather than a display terminal. In the case of microcomputers with high-speed screen refresh cycles, the ability to define and use menus is an attractive option. Menu-driven systems tend to build up a hierarchy, with each level representing a degree of detail. It is, therefore, often desirable to be able to select low level menus directly, without having to first select a number of higher level options.

2.2 Program driven

This feature is only applicable where there is a high level of user expertise and a requirement for performing certain or all DBMS functions programmatically by calling specific sub-routines. A DBMS that is composed of a set of such sub-routines is not strictly a DBMS though there are products that make such claims. A program driven DBMS is a viable option where specific and unorthodox processing is required and where conventional software is unlikely to satisfy the application requirements.

2.3 Command driven

The most common way of driving DBMS software and selecting various options is by inputting a command (or an abbreviation) followed by one or more parameters that may be optional. This method of driving the software is very flexible and easily expandable, though it can be quite unfriendly without a help subsystem and meaningful messages. The command formats should be user accessible and held externally in a file rather than hard-coded into the software. This will enable users to translate commands or to modify the format of the dialogue to suit their needs.

2.4 Help subsystem

The help subsystem is a common feature on most DBMS software and is particularly useful in the case of command driven software. A help subsystem presents the options open to the user and indicates the precise syntax of the commands. The subsystem may go further and display an example of a given command in use.

The help responses should ideally be stored in a file and be independent of the software itself and may be incorporated into the command file mentioned previously. A help subsystem, however sophisticated, should not be seen as a substitute for good documentation.

2.5 User friendliness

This particular feature of a DBMS is one of the most often talked about software characteristic, and yet, it is the hardest feature to quantify. The user friendliness of a software package is quite obviously a function of the users of the package. Their expectations, background and expertise, all contribute to colour their view as to what they perceive to be a user friendly system. It is, nevertheless, possible to add to these purely objective opinions a set of features that facilitate using the DBMS software. A few of these features, e.g. the help subsystem and menu facilities, have already been touched on, and the following represent additional desirable features:-

- Meaningful error and warning messages with references to a detailed description of the error/warning messages, with an indication of how to proceed following an error condition.

- Controlled recovery from unanticipated errors.

- Consistency in the use of the command syntax.

- Meaningful command names with suitable abbreviations.

- Free format input of commands and associated parameters, with little or no positional constraints.

- Ability to modify/amend previously entered commands without having to re-input the command.

- Comprehensive, system-provided default values for parameters that are not frequently used.

- Masking of unwanted detail from users, unless and until specifically requested.

- Minimizing the probability of accidentally selecting sensitive functions, by re-prompting and/or warning of the implications of selecting such functions. The building of tolerance to user mistakes can be taken a step further by anticipating common mistakes and thereby warning/correcting their mistakes and possibly making assumptions about what they actually intended to do.

- Ability to interrupt the execution of commands either to continue or to discontinue processing. In the event of choosing the latter, it should be possible to backtrack and rectify the results of tasks already performed so as to leave the system in a state as found prior to the execution of the selected command.

3. DATABASE DEFINITION

The software features provided by various "DBMS" products for Database definition vary quite considerably and depend primarily on the database structure supported. In the case of FMS software, file creation/modification is handled either directly or indirectly by operating system utilities and commands. In the case of hierarchical/network DBMS software the database definition process is handled via a database definition language. The database definition is, therefore, a distinct and separate activity from the creation of the physical files. In the case of RDBMS the database definition process is generally dialogue and command driven, resulting in the creation of a database definition file. The physical files are, in some cases, created and initialized at the same time.

In order to be able to evaluate various database definition procedures, a common ground must be established. To this end, the following features should serve as a guideline, though ultimately the precise method of achieving the objectives is less important than whether the objectives are achievable using a given DBMS software product. The objectives are:-

(i) The definition of a logical database structure.

(ii) The creation and initialization of the underlying physical structure.

(iii) The modification and maintenance of the physical and logical structure.

(iv) The reporting of the physical and logical structure in a coherent and comprehensive manner.

Sections 3.2, 3.3 and 3.4 deal with objectives (i) and (ii), whilst the features in sections 3.5 and 3.6 deal with objectives (iii) and (iv) respectively.

3.1 Database structure definition

The ability to define a database structure independently of the data itself is the cornerstone of database technology. As mentioned previously, the variety of database structures has resulted in the proliferation of differing structural definition procedures.

The least error-prone and most user-friendly means of capturing database structural information is through the use of system-generated dialogue and predefined commands. Regardless of the precise sequence and mechanism, the objective is to compile information on the database as a whole rather than detailed components, for example:-

(i) logical database names;

(ii) physical database names (usually system defined);

(iii) physical database characteristics, e.g. number of records and maximum/average record size;

(iv) database log file name (if logging is provided), see section 9.4;

(v) audit trail file name (if auditing facilities are provided and required);

(vi) logical links between various databases, e.g. join rules in the case of RDBMS software.

The above information is generally stored in a database definition file in an internal format. This file acts as an interface to the physical files and can be used for reporting the database structure.

It should be apparent that, in addition to the above information, the details of the database records and fields are also required and should be captured at the same time (sections 3.3 and 3.4 deal with the record and field level details).

More sophisticated DBMS software, particularly H/NDBMS, provide a database definition language (DDL) that is used to define a database schema. The schema defined in a standard text file serves as input to a database creation utility, which essentially translates all DDL statements into a database definition file. The complexity of the database creation utility is likely to vary quite considerably, though the following features are desirable:-

(i) Meaningful error messages during the schema translation process.

(ii) Physical file creation facility.

(iii) Database modification and maintenance facilities (see section 3.5).

3.2 File definition/creation

The DBMS software should ideally create and initialize the physical files using the information stored in the database definition file. Most DBMS software estimate the required file sizes using the following information:-

(i) average/maximum record sizes;

(ii) number of specified/estimated records;

(iii) hardware characteristics, e.g. disk density, number of tracks per disk, sector size.

Where possible, hardware features and characteristics should be kept independent of software features. This will ensure flexibility in the selection of storage media, e.g. hard disks, floppy disks, removable hard disks, soft sectored floppy disks, etc. It is the operating system's responsibility to present a consistent and yet flexible view of the storage media to the user. If the operating system fails to do so, the DBMS software must perform this task.

The space allocation procedure adopted by a given DBMS product is of great importance. If a given DBMS software reserves all the space that has been requested at the data definition stage irrespective of short-term and long-term storage requirements, the overall usable disk space will be effectively reduced. The alternative approach is for the software to maintain a free disk space map (or refer to an operating system maintained space allocation table) and thereby allocate disk space as and when required.

3.3 Record level definition

The database definition at this level should address the specific details that apply to database records rather than the various components of these records.

- Sort specification

Hierarchical and network DBMS software often provide facilities to define sort sequences at the record level. Imbedding record sorting specifications into the database structure enables the rapid retrieval of database records in a sorted order

without having to resort to stand-alone sort utilities. Although this feature is attractive in that it dispenses with the need for sort work files and the extra processing associated with sorting, the following points should, nevertheless, be considered:-

(i) the overheads related to maintaining sorted chains may be quite significant in the case of a larger database, particularly during data entry/modifications;

(ii) the sorted chains that are essentially pointers require additional storage space, over and above the data itself;

(iii) specifying sort sequences at this level lock the sort requirements into the database structrure thereby reducing the flexibility that is generally required during data retrieval.

- Record number/identifier specification

The ability to tag records with a unique record number/ID which can subsequently be used to retrieve the records is obviously desirable. In addition, the record numbers can facilitate the cross-referencing of records in a large report. In the case where record numbering is required only for output reports, the number assignment should be performed by the report generator.

- Default print format specification

It should be possible to associate a standard print format file with a given database. This will ensure that, unless a particular print format is specified, on retrieving data from the database it will be formatted using the default print format file.

- Variable length record specification

Most non-numeric data is variable in length and, therefore, the ability to store such data using variable length records can result in significant savings in online storage requirements. The need for this feature is proportional to the overall size of the database, i.e. if the overall size of a datbase is small relative to the available online storage space, the space wasted in each fixed-length record is less significant. In addition, it is often possible to identify fields within a record that are, on the whole, fixed in length. One can, therefore, divide a variable-length record into a fixed-length component and a

variable-length component. The two components can be stored in separate files, and depending on the database structure the two files can be joined or linked through printers. The variable component can be stored internally as a set of fixed-length records "chained" to one another. It is clear that this approach requires the retrieval of multiple fixed-length database records in order to retrieve a single logical record. In general, therefore, the need for facilities to handle variable-length data is apparent.

3.4 Field level definition

The detailed database definition is performed by defining the structure of the database records. Since records are made up of a collection of fields, the field level definitions represent the lowest level of detail for a given database. The field level micro DBMS software features that are relevant to bibliographic systems are as follows:-

- Field identifier specification

  This feature enables each field, within a given database, to be uniquely referenced. In addition to specifying field names, field mnemonics and alphanumeric field tags are also desirable, though not strictly necessary.

- Field data type specification

  The usual data types supported are: alphabetic, numeric, packed, logical, decimal and alphanumeric (alphabetic is the most commonly used data type). The mere specification of a field data type will not in itself guarantee data integrity. Type checking must be performed during data entry or modification. In addition, correct type specification, in the case of numeric data, can result in significant data compression and therefore reduce overall disk space requirements.

- Field length specification

  This feature determines the maximum length of a database field. In the case of DBMS software that support variable length records, the field length represents the maximum allowable field length and not the space actually used.

- Repeatable field specification

  This feature enables a group of fields to be clustered in such a way that they can share identical and common field characteristics. In the case of DBMS software that support only fixed length records, the maximum number of occurrences must also be specified. In this situation the space required by all the occurrences of a given field are reserved, irrespective of whether the occurrences are ever used. It is therefore apparent that repeatable fields are desirable when variable length fields are also supported. However, the use of this feature is inconsistent with the common database design practice of normalizing databases.

- Subfield specification

  This feature facilitates the clustering of fields, of various data types and characteristics, in such a way that the group of fields can be treated, when desired, as a single field.

- Field editing specification

  This feature, where provided, enables the automatic and transparent modification of data during data input and subsequent modification/retrieval, e.g. the character "/" may be stripped out of a data field (YY/MM/DD) on input and subsequently inserted on retrieval. This results in the saving of two bytes per date field, which can result in significant savings in storage requirements when an entire database is considered. This feature may be incorporated as part of the key extraction procedure, e.g. non-alphanumeric characters can be stripped out of key fields.

In addition to the above field-level features, there are additional features that enhance the data entry/modification and retrieval process. As might be expected, various commercial DBMS software implement one or more of these features in different ways. Some incorporate them as part of their data entry/modification or query facilities and others imbed the features into the database structure. The features listed below, therefore, are not necessarily part of the database definition process.

- Fast access file specification.

  This may include the specification of one or more of the following:-

  (i) online/offline key extraction;

  (ii) key length specification;

  (iii) key splitting criteria, e.g. number of keys extracted from a field containing compound data (one or more words/terms);

  (iv) noise word elimination specification (e.g. stopword list specification).

- Look-up file(s).

- Duplicate checking.

- Authority file validation.

- Vocabulary control, e.g. thesaurus file specification.

- Data entry prompting control, e.g. mandatory/optional field specification.

3.5 Database modification

The ease of database modification is an obvious and desirable feature of any DBMS software. There is, as has already been mentioned, a direct link between the type of database and the ease of modification. Database modification can occur at the logical and/or physical level. In the case of RDBMS software, that are defined using a combination of dialogue and commands, the software should either provide database modification commands or a database modification mode (with associated dialogue). This should be relatively simple as the software will be essentially modifying the contents of the database defintion file. On exiting the database modification mode or commands, the software must ensure that the definition file is consistent with the physical structure previously created. Any inconsistency must either be rectified or clearly identified.

In the case of DBMS software adopting the schema definition approach, the database modification procedure is more complicated. The steps that are generally required are as follows:-

(i) Modify the schema definition by editing the DDL statements.

(ii) Unload the contents of the database.

(iii) Purge the old data definition and create new definition by recompiling the DDL statements.

(iv) Load the previously unloaded data into the new database structure.

3.6 Data dictionary

The concept of data dictionaries has been a logical extension of the entire DBMS approach, i.e. since a database structure is made up of a great deal of detailed and fragmented data, it should be possible to establish a database containing information on the database structure itself. The attraction of this feature is clear in the case of large databases on mini and mainframe machines, where there are likely to be a great many datasets, records, fields, etc. The need for data dictionaires is much less pressing in a microcomputer environment.

Commercial microcomputer DBMS software vendors tend to use the term data dictionary in the narrow sense and refer to the data definition file as the data dictionary. A true data dictionary should include, at least, the following features:-

(i) File, record and field level definition procedures.

(ii) Facility to define print format files.

(iii) Data entry screen definition utilities.

(iv) Ability to report all the above information and also information on the physical file structure.

(v) Ability to perform consistency checks between the various components of the database, in order to maintain standardization.

It should be clear that a data dictionary must be capable of integrating all the components of a DBMS to minimize data redundancy and facilitate the retrieval of information about the DBMS.

4. DATABASE POPULATION

This section deals with the DBMS software features that are associated with capturing and storing data onto a previously defined database. The term database population has been chosen so as to incorporate both online data entry and also offline data creation.

4.1 Online data entry

The process of online data entry involves inputting data either directly into the database (updating a single record at a time) or indirectly into a transaction file. The online construction of a transaction file, and subsequent batch updating of the database, is functionally no different to the direct mode of data entry. The difference being the time at which the database is updated. As a result, this section will focus on direct online data entry.

- Prompted entry

In the case of prompted entry, the system should retrieve the necessary prompting information from either the data definition file or data dictionary. The system should prompt the user with field names in a pre-defined sequence. The following represents a set of desirable features:-

(i) User definable prompting sequence (defined during data definition).

(ii) Multiple prompting of repeating fields, including a capability to interrupt prompting and to proceed with subsequent fields.

(iii) The capability to discontinue prompting entirely.

(iv) Listing of fields, on the screen and on hard-copy device, prior to updating the database.

(v) Editing of fields prior to updating the database.

(vi) Online validation of input data against a previously-defined validation file or validation by specific user-defined routine.

(vii) Availability of commands to update the database, exit enter, clear and re-input data.

(viii) Cross-checking of fields in the same record based on data previously input.

(ix) User definable default values, that can be either over-ridden or accepted.

(x) Ability to input record number/ID with in-built checking of uniqueness.

(xi) Ability to use data from previously input record, thus facilitating data entry.

(xii) Duplicate checking of data on entry.

Prompted entry is generally more suitable for variable-length record entry. Since fields are input one at a time, data entry can continue on as many subsequent lines as are necessary. On the other hand, the ability to modify data in previously-input fields tends to be more cumbersome when fields are prompted individually.

- Formatted entry

The ability to define a screen format that can be used both for inputting new data and retrieving previously input data has become increasingly popular. This is particularly true in the case of databases where only fixed-length records are supported and where, therefore, the total maximum number of characaters (per record) that can be input is fixed and pre-defined. Screen formatting should offer the following features:-

(i) Screen editing of previously-input data by providing full cursor control.

(ii) Mode switching commands, to facilitate the switching between data entry, data modification and data retrieval.

(iii) Ability to define protected areas within a given screen, to distinguish between data that is displayed purely for information purposes and data that can be modified.

(iv) Capability of chaining screens together in order to capture variable-length data.

(v) Availability of commands to update the database with the current record and also the ability to clear and re-input data.

The essential component of screen formatting is the ability to define, in detail, the layout of the screen and to associate the relevant database fields with the input screen.

Screen definition is generally performed by using a high-level screen definition language or, alternatively, by responding to a system generated dialogue. In any event, the screen definition procedure should provide facilities for the definition of the following:-

(i) The database fields.

(ii) Labels associated with fields.

(iii) Location of fields on the screen, i.e. row and column details.

(iv) Screen title or heading.

(v) Field editing details, e.g. character stripping on entry.

(vi) Look-up file details (if not already specified at the database level).

Once a screen has been defined it should be possible to save the format in an "object" form for subsequent use. It should be clear that one of the major attractions of formatted entry is its user-friendliness and the ability to integrate data entry, modification and retrieval.

4.2 Offline data creation

There are a few occasions when alternative means of "loading" a database (other than online data-entry) are called for. The following occasions are examples:-

(i) When a structural change is made to an already established database.

(ii) When data is provided by an external and unrelated database (see section 12.3).

(iii) When data is captured on files outside the DBMS environment by non-conventional input devices, e.g. optical character readers and bar code readers.

(iv) When data is created by stand-alone programs.

The precise mechanism of loading a database, e.g. through stand-alone utilities or user-written programs, is less significant than the desirability of having features enabling the offline creation of data.

5. DATA MODIFICATION

This feature should be considered as an extension of the data entry process since the input and editing of a new record is functionally similar to the retrieval and editing of an old database record.

5.1 Online data modification

The features that are identified and expanded in section 4.1 apply equally well to online data modification. In addition, an interface to the data retrieval sub-system must exist so as to facilitate the retrieval of previously input data. In general, users do not make a clear distinction between the data entry, modification and retrieval activities. Instead, they view them as different aspects of the same activity. It is, therefore, desirable to integrate, as much as possible, these discrete activities into a single and coherent process. The integration of all the desirable features associated with data entry, modification and retrieval is unrealistic in a microcomputer environment, primarily due to the run time overheads. It is, nevertheless, desirable to, at least, combine a subset of the desirable features from each of the three activities.

5.2 Global data modification

The ability to globally modify the contents of one or more records requires the following:-

(i) The specification of the data selection criteria.

(ii) The specification of the data modification criteria.

The attraction of applying the same data modification criteria to a group of records without having to modify each individual record separately should be apparent. The overheads associated with this feature can be quite considerable and particularly so in a multi-user/multi-tasking environment.

5.3 Data deletion

In addition to modifying the contents of records already present in a database, it should also be possible to "flag" records as being logically deleted. This feature will effectively prevent the deleted records from being accessed by users.

An extension of this feature is the ability to phyiscally delete records from the database, thereby recovering valuable disk space. This feature may be incorporated into a database maintenance utility or sub-system, enabling perennial structural reorganization and optimization.

6. DATA RETRIEVAL

The data retrieval features of DBMS software are the most significant features as far as bibliographic applications are concerned. The ease of data retrieval in conjunction with output generation is usually the primary reason given for selecting a DBMS to solve the problems associated with information management. As mentioned earlier, the DBMS software architecture has a major bearing on the flexibility of data retrieval. There are, however, other specific features that greatly enhance and, therefore, facilitate the selection and manipulation of data for subsequent output.

6.1 Online searching

The structure of bibliographic data, and non-numeric data in general, is such that a variety of specific tools are needed in order to select the required data for subsequent processing. The database size is a prime factor in the number and nature of the required tools, i.e. the larger the database, the greater the need for search features. The problem stems from the need to extract as much desirable data as possible from the database without, in the process, picking up "noise" data. Such unwanted data tends to cloak the desirable information with a great deal of peripheral and unwanted data, thereby complicating the search strategy. This section lists a set of features that are desirable and which facilitate the process of online searching.

- Free text searching

This is the least efficient method of locating data and consists of scanning fields character by character, attempting to locate a match against the required character string. The suitability of this option depends on the size of the database to be scanned. In the case of small databases or subsets of large databases this feature may be quite adequate for searching.

- Searching using a range of record IDs

The ability to select a range of records based on a numeric sequencing number is highly desirable. The availability of this feature enables the selection of subsets of the database irrespective of the data within the desired record range.

Although most DBMS software assign numeric sequencing numbers to records at data entry time, these numbers are not always user accessible. In cases where the numbers can be accessed, they

should be used with care. This is due to the fact that the numerical order of the internal sequence number may not necessarily correspond to the chronological order of the records.

Furthermore, the sequence numbers may be modified regularly by utilities, thereby reducing their usefulness as a means of identifying records. In situations where the sequencing number is user definable, adequate duplicate checking is required.

- Word and term processing/key generation

As mentioned earlier, there are many applications in which free-text searching is inadequate due to the volume of data. In such situations, there is often a requirement for searching at the level of words or combination of words, i.e. terms. The ability to perform word and term processing efficiently requires special data structures and the capability to split a series of words or specially delimited terms into separate keys. The required data structures and the way in which data redundancy is handled is dealt with later.

- Boolean and nested search logic

The ability to translate a search strategy into an effective search request requires Boolean logic. The use of Boolean logic, e.g. AND, OR and NOT is generally available on most DBMS software, and when used in conjunction with brackets to establish a hierarchy of operation, the data selection process is made considerably easier.

- Back referencing

The effective searching of a bibliographic database using a defined search strategy entails expanding and contracting the size of the data selected. Most search strategies are made up of multiple steps, each step selecting and operating on a subset of the database. The ability to reference previous search results (hits) and to add data to or exclude data from previous hits, is a highly desirable feature. This feature obviously requires the ability to create and save hit lists.

- Storage and retrieval of search strategy

Search strategies are often multi-stepped and can be long-winded. The ability to save a search strategy in a standard external text file for future use is a convenient feature. In addition, the availability of this feature enables the building up of search profiles outside of the DBMS environment.

- Special features

In addition to the above features, there are further special features that are uncommon in microcomputer DBMS software. These features can enhance the user friendliness of the product, and are generally available as extensions to the word processing features mentioned earlier. They include adjacency searching, left/right/embedded truncation, and proximity searching. There are, no doubt, additional features not mentioned or variations on themes already touched on.

6.2 Batch extraction

The ability to search a database in batch mode using a previously created search profile is a desirable feature. Most microcomputer DBMS functions tend to be I/O bound and therefore any feature that reduces human intervention is likely to improve the overall throughput of the system. The need to perform multistepped database searches on an entire database, followed by sorting and searching, can require a significant amount of machine resources and, therefore, time.

The need for substantial operator intervention at various stages of performing multi-stepped tasks results in periods of intense activity punctuated with periods of little or no activity. The ability to consolidate many discrete activities into command or job files minimizes human intervention and is a common feature in larger machines, and is becoming increasingly popular in microcomputers.

6.3 Fast access files

The ability to retrieve data by specifying key values is highly desirable in bibliographic applications where there is a large emphasis on online data retrieval. The level of sophistication of the retrieval facilities depends on the types of fast access files supported by the DBMS software. As might be expected, each software producer has adopted one or more conventional file access method and has in some way modified it to serve their own needs. This has resulted in the availability of a variety of access methods, some conventional and others hybrid. The various types of fast access files and the features offered by them are dealt with in Appendix C.

7. ARITHMETIC COMPUTATION

Traditionally, the demand for micro DBMS software has been from the financial community and, therefore, the commercial software products that are currently widely available tend to support a variety of features geared to arithmetic computation. Many such features have little practical use in bibliographic applications, though the following list of features can be of value on certain occasions:-

(i) Support of basic arithmetic functions, e.g. addition, multiplication, subtraction and division.

(ii) Horizontal computation, i.e. calcuation of values based on the contents of fields within the same record.

(iii) Vertical computation, i.e. calculation of values across multiple records.

(iv) Use of temporary variables, symbolically identified, for generation of intermediate values and used in subsequent calculations.

(v) Total and subtotal generation.

(vi) Saving the results of computations back into the database.

8. OUTPUT GENERATION

The term output generation encompasses a variety of features. These features are all associated with formatting data that will have already been retrieved and possibly sorted. The desirable or required features for output generation vary depending on whether data is to be displayed on the screen, formatted for a report, or merely transferred to a new disk file in a new format. The process of output generation can be divided into three broad categories: Print format definition, Output creation, and Output redirection. Each category will be dealt with in turn.

8.1 Output creation

The ability to define the format of a report based on detailed specifications, without resorting to programming is an obvious attraction. However, all report generators attempt to strike a balance between flexibility, ease of use, and level of complexity. The report generation capability of a given DBMS software package, and therefore its ultimate usefulness, depends on one or more of the following factors:-

(i) Sophistication of the report generation utility.

(ii) Programming language interface (see section 10.3).

(iii) Macro generation capability (see section 10.4).

(iv) High level procedural language subsystem.

The availability of a programming language interface enables the production of the most specific and complicated reports, though this approach should be considered as a last resort, the main drawbacks being:-

- The need for programming expertise.
- The relatively long development time.
- The need for software modifications as and when database changes are made.

Most commercial report generation software construct print format files using responses to system generated prompts and predefined user-input commands. The Print file, held in an

internal format, is interpreted at run-time produing the specified output. The ability to save a print format file for future use is an obvious attraction since the print format file can be used in the routine production of the same report, without having to define the print format each time. In addition, the print format file can be used as a template for producing new though similar reports. The following represents a list of additional features that are desirable for output creation:-

(i) Capability of modifying previously-defined print format files.

(ii) Ability to "compile" print format files into stand-alone print programs that can be executed under the control of the operating system and without the DBMS run time overheads.

(iii) Facility to generate default print format files using information already captured in the data definition file or data dictionary.

(iv) Ability to spool the output in order to increase the overall system productivity (this feature depends on the support of multi-tasking/concurrency by the operating system and/or the availability of stand-alone printer spooling hardware).

8.2 Print format definition

The previous section referred to the ways in which print format files should be constructed and used. This section addresses the specific features that contribute to the definition of print formats used in the reporting of bibliographic data. The number and precise nature of the available features will determine the complexity of the reports ultimately produced. The definition of a print format should be handled at three levels: page, record and field; each will be discussed in turn.

- Page level formatting

The following features establish the general layout and appearance of a report irrespective of the data held within the database:

(i) Printing of page headings or report titles.

(ii) Printing of a report date.

(iii) Ability to define the number of lines per page.

(iv) Printing of page numbers and specification of the starting page numbers.

(v) Printing of page footings.

(vi) Columnar formatting specification to force printing of the data within specific column ranges.

(vii) Performing left and right margin justification.

- Record level formatting

The following features apply to individual records within a report page:-

(i) Vertical spacing between records.

(ii) Spacing between records printed on the same line.

(iii) Record numbering.

In the case of DBMS software that support variable length records, wrap-around capability is essential since the truncation of records following a specific and fixed number of characters could result in the loss of meaningful data.

- Field level formatting

The following features determine the position and form of individual fields within a record:-

(i) Optional printing of literals based on the contents of fields.

(ii) Clustering of repeatable fields within a report page.

(iii) Printing of fields in a given location by specifying an absolute print location.

(iv) Editing of specified characters prior to printing.

(v) Associating control characters with selected fields for special printing (e.g. bold character printing, shadow printing, proportional spacing, etc.).

8.3 Output redirection

The term "printing" a report is often used loosely and does not necessarily imply the production of a hard-copy report. The ability to redirect a report to different devices greatly extends the usefulness of the report generation faciltiies of DBMS software.

The ability to display reports on the screen has the following advantages:-

- Allows online browsing of reports for rapid proof-reading.
- Enables quick checking and correction of report format specifications.
- Complements the on-line Query facility.

Although the ability to display reports is an obvious attraction, the capability of directing reports to offline storage media enables the following functions to be performed:-

(i) Transferring reports to disk/tape for archive purposes.

(ii) Creation of reports at a faster rate and printing of reports during off-peak hours (if spooling is not available).

(iii) Additional processing on reports, e.g. photocomposition or editing of reports prior to creating hard-copy output.

(iv) Using output of one system as input to another system (see section 12.3).

(v) Transferring report files to remote users, using data communication hardware and software (see section 12.3).

8.4 Sort/merge

Most DBMS software provide sort capabilities with a wide range of sophistication. It goes without saying that sorting is an essential requirement for bibliographic systems and the ability to sort on a primary key and at least one other alternate key is a minimum requirement. Sort features are generally implemented as stand-alone utilities that are called prior to either screen or

print formatting. Sort utilities tend to make substantial demands on computer memory, and the precise mechanism by which a DBMS performs sorting relates to the available main memory and the nature of the files to be sorted. The detail of various sort algorithms and strategies is well documented but it is, nevertheless, worth considering the practical implications of some common implementations. On the whole, sort utilities require work files that are at least as large as the file to be sorted. This requirement is generally impractical, and particularly so when the source file is large and spans more than one mounted disk. In these situations the sort utility must fragment the source file into manageable subfiles and perform the sort on these subfiles. Ultimately the subfiles have to be merged and, if necessary, logically spanned across multiple disks and/or devices. It should be apparent that, in a micro DBMS environment, the availability of temporary offline storage space is of great importance for performing functions such as sort/merge. In order to reduce the size of the required work space, DBMS software producers have adopted various approaches. The most common is to maintain the data (or rather pointers to the data) in a sorted order. This is achieved using sorted index files and/or sorted chains, which are dynamic in nature and are up-dated at data entry time or whenever data is modified. This approach imposes overheads both in terms of space (for pointers) and response time (for data entry and modification). The major drawback, with freezing the sorting requirements of a given application into the DBMS structure, is that responding to changes in sort requirements in the future becomes non-trivial. In practice, the need for flexible stand-alone sort/merge utilities is an indispensible feature for micro DBMS software used in bibliographic applications.

9. DATA INTEGRITY

The primary objective of any DBMS software, whether in a microcomputer or in a much larger mini/mainframe environment, should be the maintenance of data integrity. The integrity of a database can be compromised in many ways. The features provided by a given DBMS software package should address the following potential problem areas:-

(i) Unauthorized access of a database by malicious users.

(ii) Accidental corruption of data through user mistakes.

(iii) Contention between multiple users of the same database.

(iv) Hardware failure and software bugs.

The features documented in the following sections should attempt to minimize the impact of data corruption from one or more of the above sources.

9.1 Database security

The issue of database security is much more pressing in a large multi-user environment with remote access capabilities. In the case of single user micro DBMS applications, the issue of database security is, nevertheless, relevant though much less pronounced. The availability of the following features is desirable:-

(i) Password protection of the physical database files.

(ii) Password protection of the database definition sub-system.

(iii) Definition of a user hierarchy to distinguish between types of users.

(iv) Definition of a set of capabilities and associating them with users or user groups.

The ability to prevent users or groups of users from accessing certain fields within a database by using "project lists" or special screen formats is an obvious attraction. Similarly, the ability to define record exclusion criteria can be used to exclude large portions of a database from users.

9.2 User passwords

The ability to define user passwords is the most common security feature currently available. This is due to the relative ease of providing this feature. However, the effectiveness of user passwords is questionable since the degree of security provided is directly related to the ability users have in protecting the secrecy of their selected passwords.

9.3 File locking

An effective file locking strategy is essential in the case of DBMS software that are capable of executing in a concurrent or multi-tasking operating system environment. In the case of a multi-user environment, a DBMS will not be able to perform adequately without file and record locking features that resolve contention between users during data entry/modification.

9.4 Transaction logging/recovery

This feature is widely available in large DBMS applications where database up-dates are grouped into discrete transactions. The transactions are first copied to disk or tape prior to being applied against the database. The file containing the copies of transactions serves as a transaction log which can be used to recover data that may be lost in the event of hardware or software failure. The overheads associated with transaction logging balanced against the potential impact of any loss of data should be considered when determining the need for such features.

9.5 System backup

In order to minimize the impact of data corruption or loss, there is really no substitute for regular system backups of the physical database files and database structure. The precise details of system backup are not really significant though any DBMS software feature that facilitates this routine activity is highly desirable.

10. UTILITIES AND SPECIAL FEATURES

The features listed in this section can be thought of as being optional. The availability of some of these features, however, can enhance a given commercial DBMS package to the point of reducing the significance of other more obvious shortcomings.

10.1 Software configurability

The diversity and variety of microcomputer hardware and operating system software has presented commercial DBMS software producers with the problem of selecting the most promising hardware and operating system combination for their particular DBMS product. In order to reduce the impact of an incorrect choice and, at the same time expand their their potential market, they have resorted to prodividng configuration utilities to dealers and users. These utilities ensure that, as the run-time environment of a DBMS product changes, the software package can be adapted accordingly. The need and, therefore, desirability of this feature is likely to decrease once the microcomputer market stabilizes and the inevitable domination of the market by a few vendors has been completed.

10.2 Disk file chaining

The storage capacity of microcomputer flexible disks varies quite considerably depending on, for example, the physical characteristics of the disks and the disk formatting strategy adopted by the operating system. Bibliographic applications tend to make major demands on storage space, not merely because of the total number of database records required, but because of the size of individual database records. The ability of databases to overflow one physical disk is, therefore, highly desirable. In the absence of such a feature, the limiting factor on the size of a database will not be the maximum number of DBMS records allowed, but, rather, the availability of sufficient storage space.

The overheads associated with establishing and maintaining a master disk directory to keep track of the constituents of physical files (fragmented across multiple disks) is, on the whole, justifiable.

10.3 Programming language interface

The flexibility offered by programming languages has resulted in interfaces being built into DBMS software so as to enable specialized and user-specific functions to be performed. Although this feature is generally desirable, its practicality is limited by the availability of experienced programming personnel. In addition to the required level of programming expertise, once a user-specific module has been developed, it will remain static unless it is kept continuously up-to-date with the DBMS application. It is for this reason, as has already been mentioned, that screen and report formatting should be performed using features provided by the DBMS software rather than by stand-alone programs.

10.4 High-level procedural language subsystem

Command driven commercial DBMS software provide a variety of features that encourage the use of commands in combination with one another. The ability to define a set of high-level commands in such a way that they work in conjunction with one another to perform specific tasks is a highly desirable feature. This provides a high-level programming capability without many of the disadvantages of programming using a high-level language. The advantages offered by this feature are:-

(i) Relatively insignificant development time.

(ii) Flexibility of performing user-specific functions.

(iii) Maintainability of the routine.

(iv) Low level of expertise needed to perform reasonable complicated tasks.

In some DBMS software, commands are provided specifically to satisfy user-specific needs and are incorporated into a high-level language subsystem.

10.5 User-defined commands/macro generation

In addition to the features already mentioned, the ability to define new commands is quite desirable. This feature can take many forms, but essentially involves passing parameters to

system-defined commands using symbiotic substitution. The attraction of this feature is that commands can be combined in new ways and parameters can be passed dynamically to the commands at execution time.

Using this feature in conjunction with features already mentioned can present users of a DBMS package with options that may not have been apparent at first glance.

10.6 **Multi-user capability and local area networking (LAN)**

The advantages offered by DBMS software running in a multi-user environment have been dealt with at length. However, a variation on this theme is the concept of local area networking. The ability to cluster a group of stand-alone microcomputers around a hard disk device, each communicating with the storage device through a communication network (e.g. token passing ring network, bus or star network) can offer the following advantages:-

(i) Standardization of data through a centralized database.

(ii) Sharing of files by multiple users via single-user microcomputers.

(iii) Availability of larger amounts of storage than would normally be available in a single-user configuration.

As might be expected, the technical means of providing the above facilities are numerous, though the need for a network traffic manager and comprehensive file locking and security is a prerequisite.

10.7 **File compatibility**

The physical file structure of a DBMS is generally of little interest to users of the software and yet the details of the physical structure have a major bearing on the following:-

(i) Off-line data creation (see section 4.2).

(ii) Database loading and unloading (see section 10.8).

(iii) Downloading data from external databases (see section 12.3).

In general, the simpler the physical file structure the easier it is to perform the above tasks. If the volume of data transfer to and from a given DBMS application is likely to be high, the need for simple data structures becomes even more desirable. This is due to the overheads associated with having to reformat files prior to performing the data transfer. The issue of standardized data exchange formats will be dealt with, briefly, in section 12.3.

10.8 Database loading and unloading

A distinction must be made between this feature and the download/upload capability of a given database, which is dealt with in section 12.3. In order to be able to reorganize the physical and logical structure of an established database, the DBMS software must provide features that enable the data already input to be unloaded, and subsequently loaded back, onto the new database structure. These features, therefore, provide the capability of reorganizing data already present in the database rather than exchanging data between databases. Having said this, it is desirable to combine the load/unload and download/upload features into a single utility. Unfortunately, this is less practical than it might seem, since the format of a file containing unloaded data is rarely suitable for exchange purposes. This is particularly true in the case of N/HDBMS software where information on chain headers, internal pointers, etc., is imbedded into the unloaded data.

11. SOFTWARE AND HARDWARE ENVIRONMENT

This section refers to additional factors that need to be taken into consideration when evaluating a given commercial DBMS package. It goes without saying that "paper" evaluations are, at least, only as good as the information provided by the software producers and vendors. Aside from the inherent bias built into such information, DBMS software never function in a vacuum isolated from other system components, e.g. hardware/firmware and operating system. It is, therefore, necessary to consider, as best one can, the environment in which a DBMS software is to function, in addition to the features offered by the DBMS itself.

11.1 Software specifications

The following list represents a set of software specifications that are relevant either directly or indirectly to the overall performance and capabilities of a microcomputer system. The term "system" has been used to incorporate hardware and operating system software constraints, in addition to limiting factors introduced by the DBMS software.

- Maximum number of database records allowed.
- Maximum number of files allowed per database (where applicable).
- Maximum number of fields allowed per record.
- Maximum number of characters allowed per database record (if not variable).
- Maximum number of characters allowed per database field (if not variable).
- Maximum number of files that can be open/accessed simultaneously.
- Maximum number of fields that can be up-dated simultaneously.
- Maximum number of user-definable variables (where applicable), and maximum memory available to temporary variables.
- Maximum number of predefined screen and print formats

- Maximum size of sort files.
- Maximum number of keys that can be extracted.
- Maximum size of each individual key.

11.2 Hardware specifications

It is becoming increasingly difficult to ignore hardware specifications when evaluating DBMS software. This is particularly true when the issue of performance is raised. The following items have an impact on system performance to varying degrees:-

- Minimum memory requirements versus maximum size of memory available.
- Memory expansion capability.
- Size of CPU data bus.
- Size of smallest directly addressable memory location.
- Size of I/O buffers.
- Disk I/O speed.
- Disk types and media characteristics, e.g. hard disk, flexible disks.
- CPU clock frequency.
- Serial versus parallel I/O ports.
- Maximum number of disks supported.
- Disk rotational speed.
- Disk latency.

11.3 Software portability

The issue of software portability has always been of concern to producers, vendors and users alike. Rarely has this concern been translated into action that would produce realistic and tangible results. With the dramatic fall in the cost of micro

systems, the issue of portability has been tackled with greater urgency. This has been the direct result of lower profit margins associated with new software products (at least initially) and their relatively short life expectancy. Portability can be achieved at the following levels:-

- Application level, e.g. through the use of portable physical database file structures.

- Operating system level, e.g. through the use of portable software development languages such as UCSD-Pascal and "C" etc.

- CPU level, e.g. through the use of operating systems such as UNIX that are portable across multiple CPU types.

The issue of portability is, however, complicated by the degree and extent of dependency between each of the above-mentioned layers.

11.4 Software support and documentation

The need for comprehensive, accurate and easy-to-follow documentation cannot be stressed enough. It is, therefore, unfortunate to find that the most useful and technically sound DBMS software often have the worst documentation. In addition to documentation, both technical and user-oriented, the following points should be considered:-

- Availability of software support.

- Availability of software maintenance contracts.

- Accessibility to new releases of the software.

- Availability of computer-based learning tools including sample databases and examples.

12. DATA TRANSFER

With the dramatic increase in microcomputer hardware and software capabilities, the concept of distributed processing has been taken a considerable step further. The dramatic developments in data communication technology and terminal emulation software have reduced, significantly, the traditional impression of micro DBMS applications as being "isolated islands of information". The features that have contributed to the viability of transferring data to and from a microcomputer are, primarily, hardware, operating system and data communication software features.

12.1 Device handling capability

This feature has little to do with the DBMS software itself but, rather, relates to the ability of an operating system to handle a variety of storage devices, e.g. flexible disks, hard disks and tape drives. However, the issue of device handling goes beyond the capability of handling a variety of storage devices and can be extended to cover a range of input devices, e.g. bar code readers, optical character readers. It is, nevertheless, the function of the operating system to define and maintain a standard and uniform interface between devices and application software, in such a way that the idiosyncracies of each individual device become transparent to the system user. The details of the various types of device handlers are, however, irrelevant in this context.

A DBMS software package that has been designed to function only with a specific type of flexible disk and running in an operating system/hardware environment with little or no device handling capability may, nevertheless, satisfy short-term needs. However, with little prospect for future expansion, the system as a whole will undoubtedly be handicapped and offer little flexibility in transferring data to and from other systems.

12.2 Asynchronous data communication

The substantial growth in the data communication industry is a reflection of the ever-increasing need to transfer data between computers. The three essential components needed for data communication are:-

(i) Data communication software (required for terminal emulation and protocol handling).

(ii) Signal modulator-demodulators or modems (perform digital-to-analog and analog-to-digital conversion for transmission over public telephone lines).

(iii) Asynchronous data communication ports (necessary in order to establish the physical connection between the computer and modem).

Data transfer is generally performed outside of the micro DBMS environment and under the control of the data communication software. The full integration of DBMS and data communication software is, however, merely a question of time.

12.3 **Download/upload capability**

The ability to download data from a host (mini/mainframe) computer onto a micro DBMS and vice versa (uploading data) has created the following opportunities:-

- Microcomputers serving as front-end data entry systems for large databases resident on host computers.

- Microcomputers functioning as intelligent data retrieval nodes, enabling direct and rapid transfer of large volumes of data for subsequent browsing. This would be a cost-effective method of searching remote databases (the copyright issues and implications are, as yet, unclear).

- Using micros as back-end processors to larger machines enabling, for example, the editing/summarization and local printing of reports generated on the host computer.

In addition to the above, a potential exists for exchanging information using a variety of storage media, e.g. flexible disks, cartridge/reel-to-reel tapes and removable hard disks. In order to realize this potential and develop a practical and realistic data exchange procedure, standards must be established in the area of the storage media and data exchange formats.

APPENDIX A

High-Level Functional Specifications

A.1. INTRODUCTION

With the dramatic increase in the processing capabilities of microcomputers and a corresponding reduction in their cost, fairly sophisticated applications which were once restricted to mini and mainframe computers can now be implemented on micros. This trend has opened the door to many applications and will continue to do so for the foreseeable future. There are, however, some major considerations which have to be taken into account when considering using microcomputers for primarily bibliographic applications.

This section will define the functional specifications of a "typical" microcomputer-based information storage and retrieval system. It is not intended to be used in the design of microcomputer-based bibliographic systems but rather to establish a reference point and a basis for further study in the identification of suitable, commercially available software. It is appreciated that in identifying such software there are unlikely to be perfect solutions. The identification process will involve a series of trade-offs and compromises. The detailed parameters used during this process will be governed by the actual requirements of a given documentation centre.

A.2 SCOPE AND OBJECTIVES

It is no doubt apparent that before attempting to establish generalized guidelines on how to select commercial software, we need to be clear about what is meant by a "typical" small documentation centre. To this end, data flow diagrams (DFD) and function charts have been used to identify what functions are actually performed in a small documentation centre and how information flows between these functions. In order to make this document a practical one, certain assumptions were made about the nature of the data and the types of storage areas used. The data flow diagrams, therefore, represent physical data flows and not logical* ones.

In order to emphasize the practicality of this exercise, it was decided to select an IDRC in-house project for detailed study. The DEVSIS project was finally selected as it represents a medium/small decentralized information centre. The DEVSIS project, at the moment, manages information on approximately 7 000 titles, making 600 additions per year.

The objective, however, is not to document the DEVSIS system from a functional perspective but rather to identify general functions which are applicable to many such systems. Not all functions are likely to be performed by any given small information centre, though various permutations of these functions are no doubt performed either manually or automatically.

---

* Logical data flow diagrams do not make a distinction between a filing cabinet and a computer disk.

A.3 FUNCTION STRUCTURE CHARTS

This section presents the various functions performed in a small information/documentation centre in a hierarchical fashion from the highest level to the required level of detail for this study.

The function charts serve the purpose of identifying the various processes and their hierarchical dependencies, and in addition represent a graphical table of contents for the data flow diagrams in the next section.

The highest level function i.e. "PRODUCE BIBLIOGRAPHY" represents a broad set of functions which for any given application may include the production/maintenance of one or more of the following:

Bibliographies
Indexes
Authority files

Each function can in turn be divided into sub-functions as shown in figures 3.1 thru 3.4.

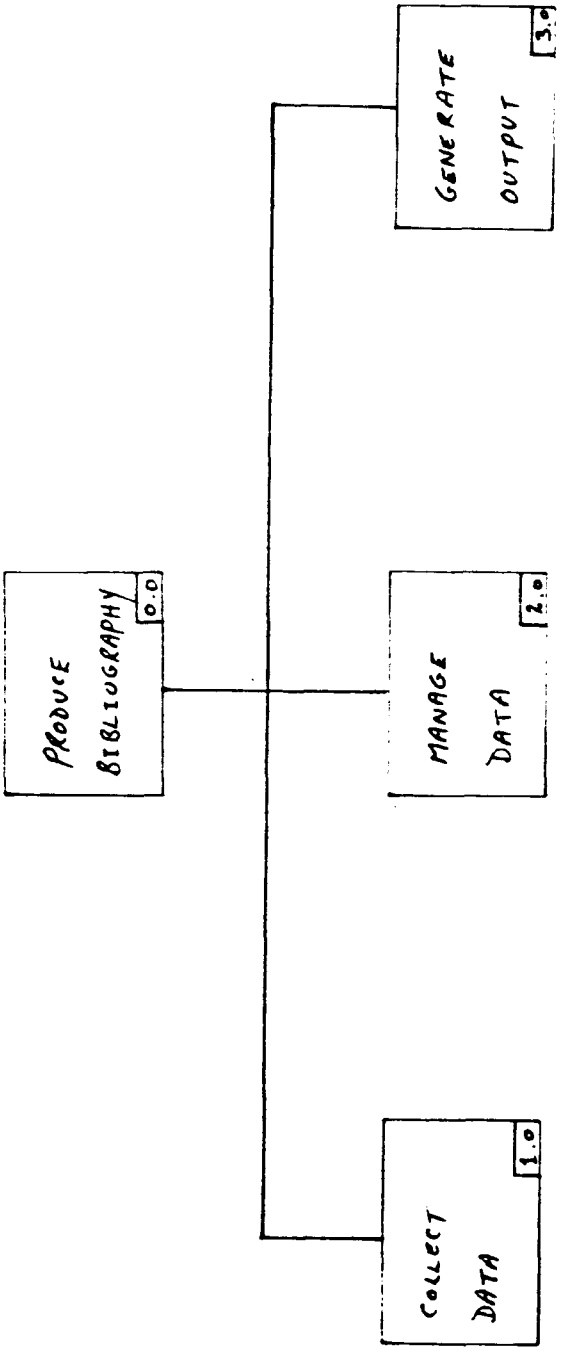

PRODUCE BIBLIOGRAPHY 0.0
COLLECT DATA 1.0
MANAGE DATA 2.0
GENERATE OUTPUT 3.0

FIG. 3.1

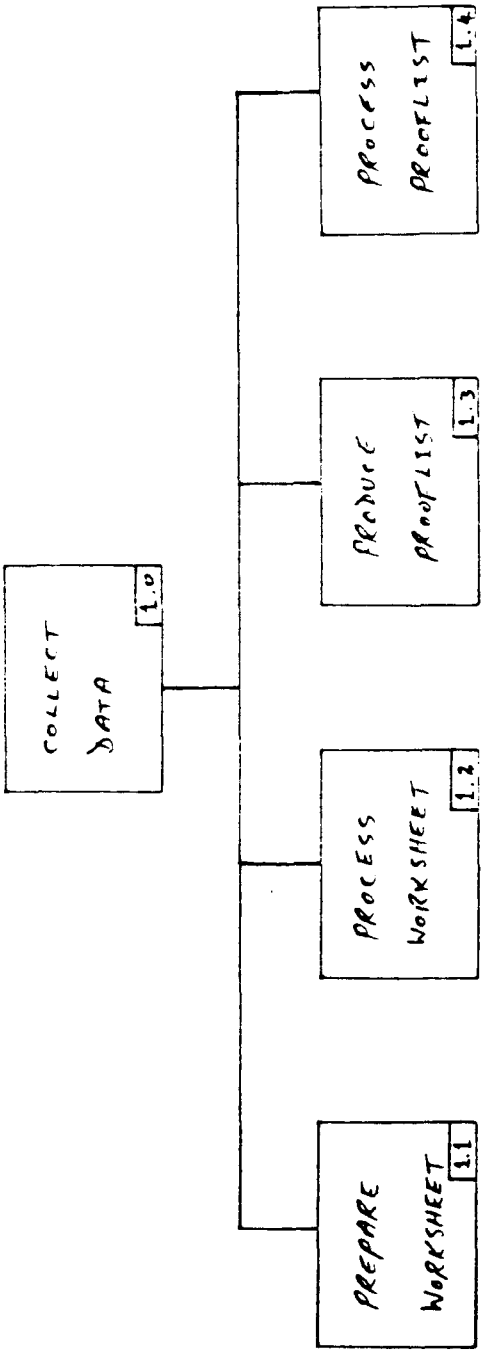

COLLECT DATA
1.0
PREPARE WORKSHEET
1.1
PROCESS WORKSHEET
1.2
PROCURE PROFILIST
1.3
PROCESS PROFILIST
1.4

FIG. 3.2

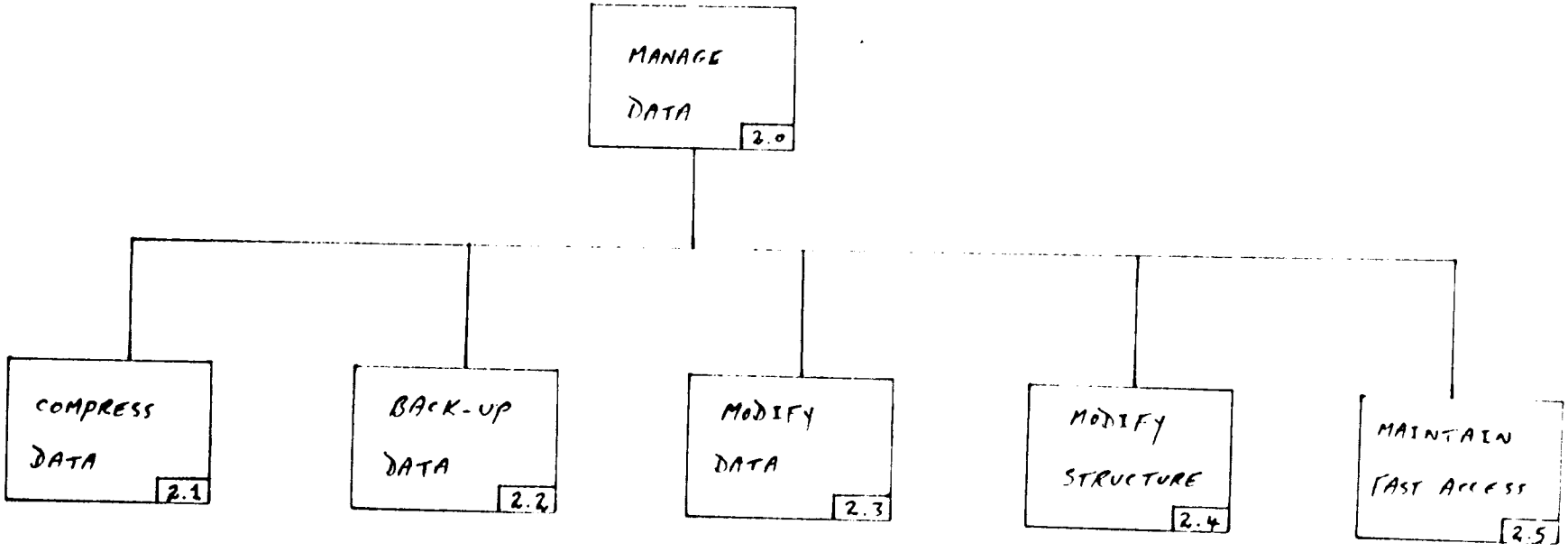


FIG. 3.3

. FUNCTION CHART : MANAGE DATA

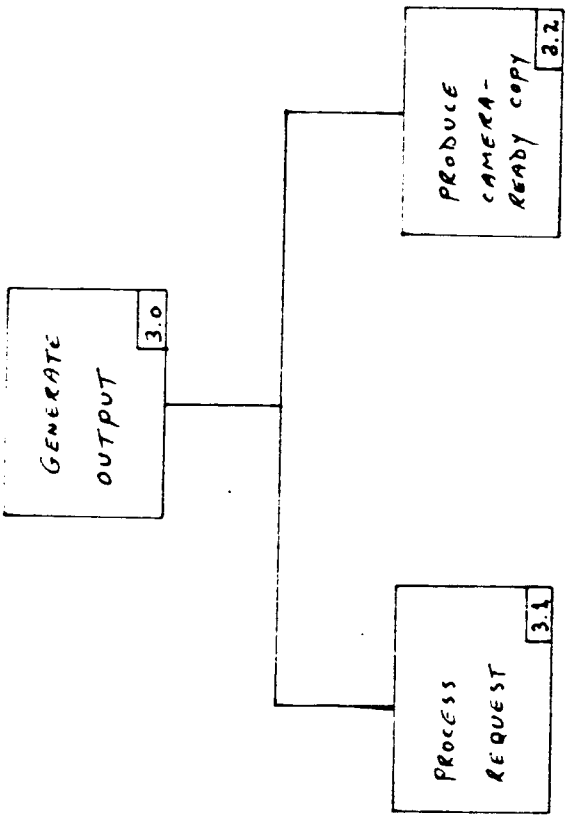
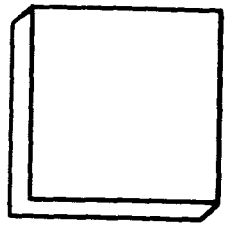


FIG. 3.4

FUNCTION CHART: GENERATE OUTPUT

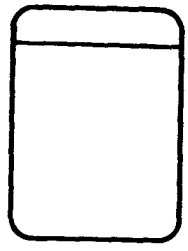
A.4 DATA FLOW DIAGRAMS

The diagrams in this section show the flow of data among the various processes, the related data-stores, and the entities which are logically external to the function being described. Each diagram corresponds to a function identified in Section 3. The following symbols are used in the data flow diagrams:



External Entity:

This symbol represents any logical class of things or people which can be considered a source or destination of data, which is External to the function.

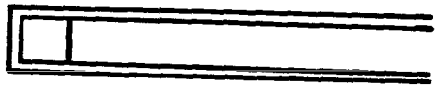


Process:

This symbol represents a function or activity carried out on data as it moves through the system. The function reference number is inserted at the top of the symbol. This is the same number used for the functions in the function structure charts of Section A.3

Data store:

This symbol is used to show non-automated data at rest between processes. These data stores may be in the form of files, forms, catalogues. Data stores are viewed as conceptual warehouses of data.



Computerized Data Store:

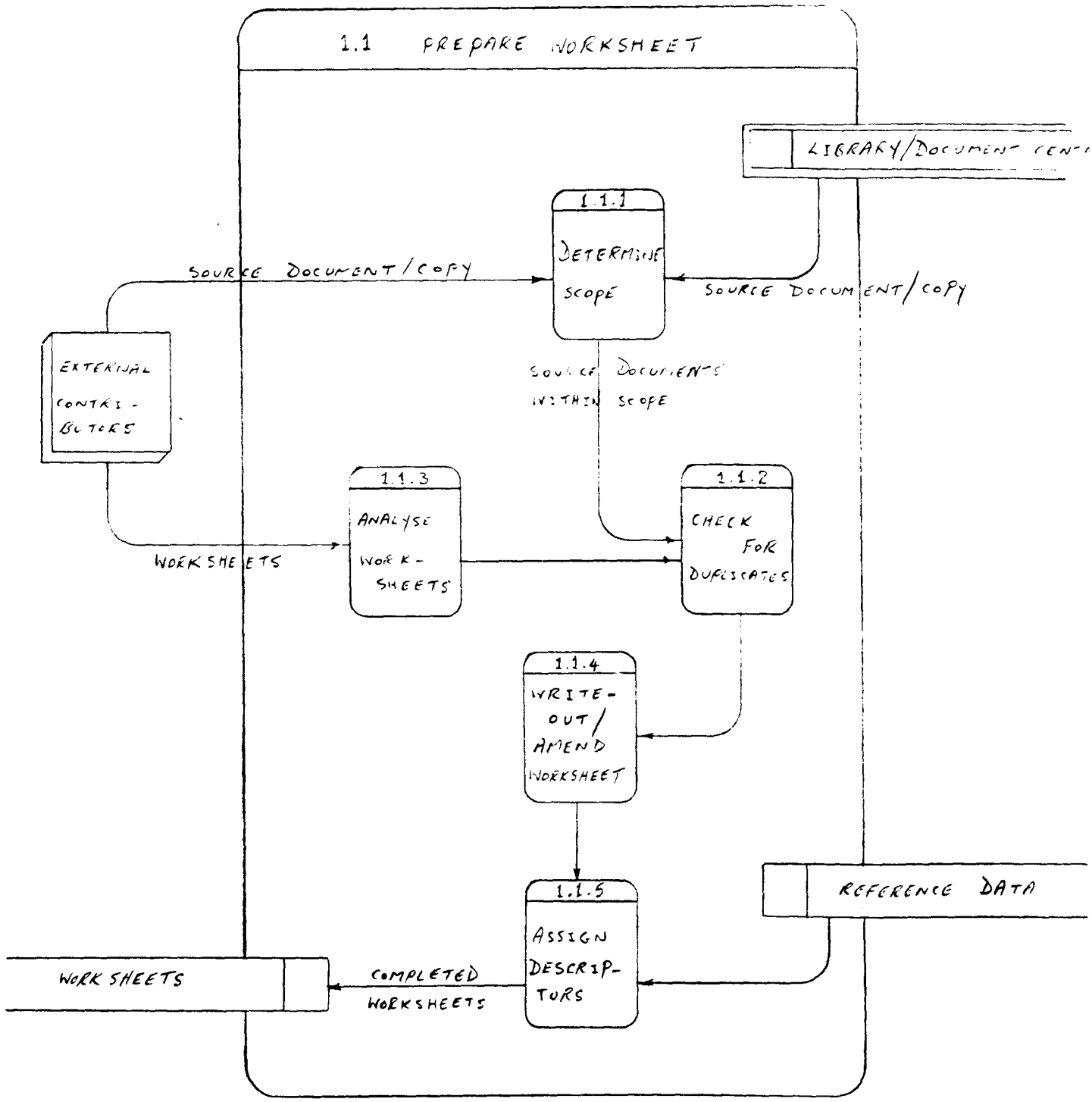
This symbol represents a data store which is automated.

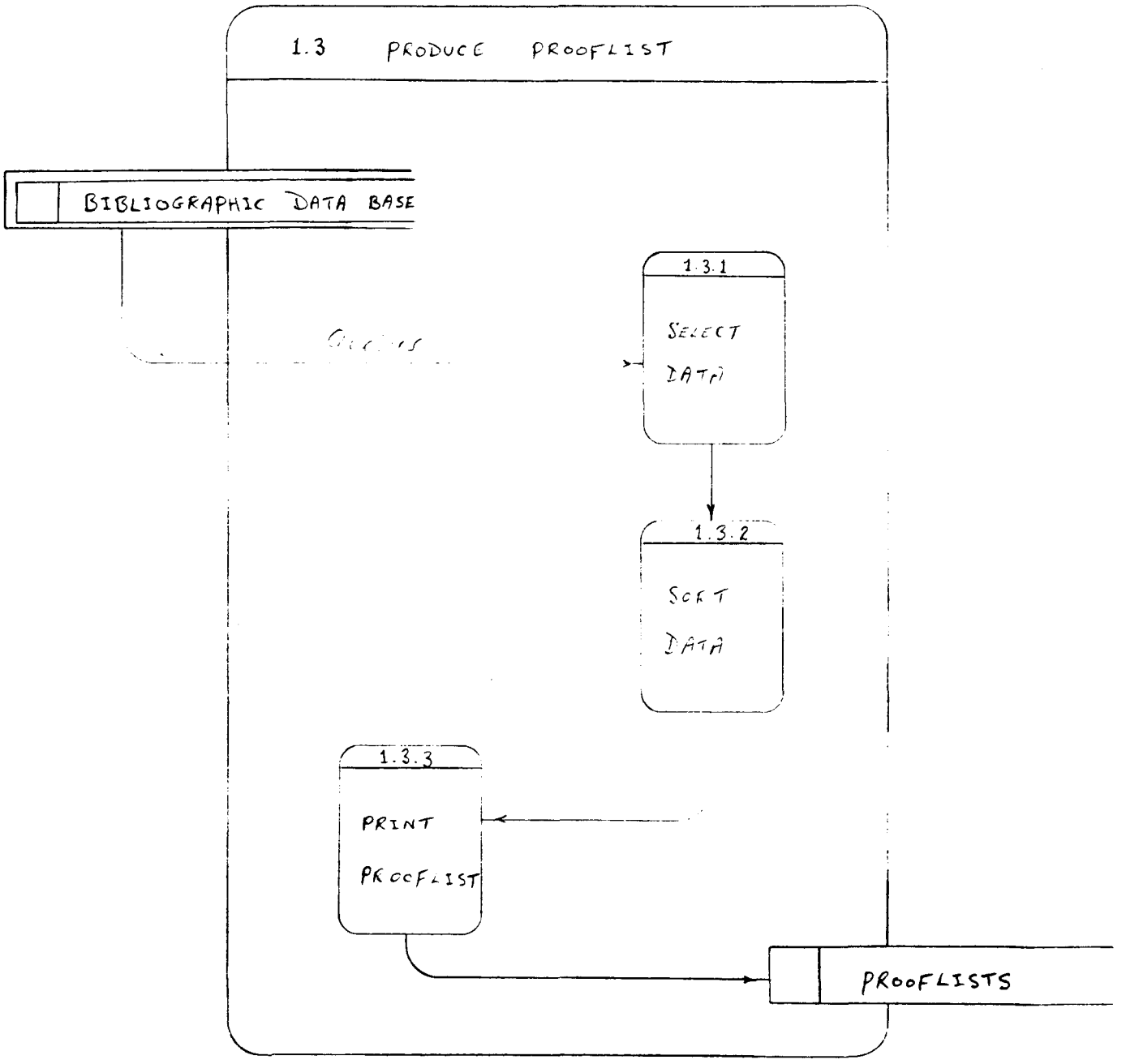
Data Flow:

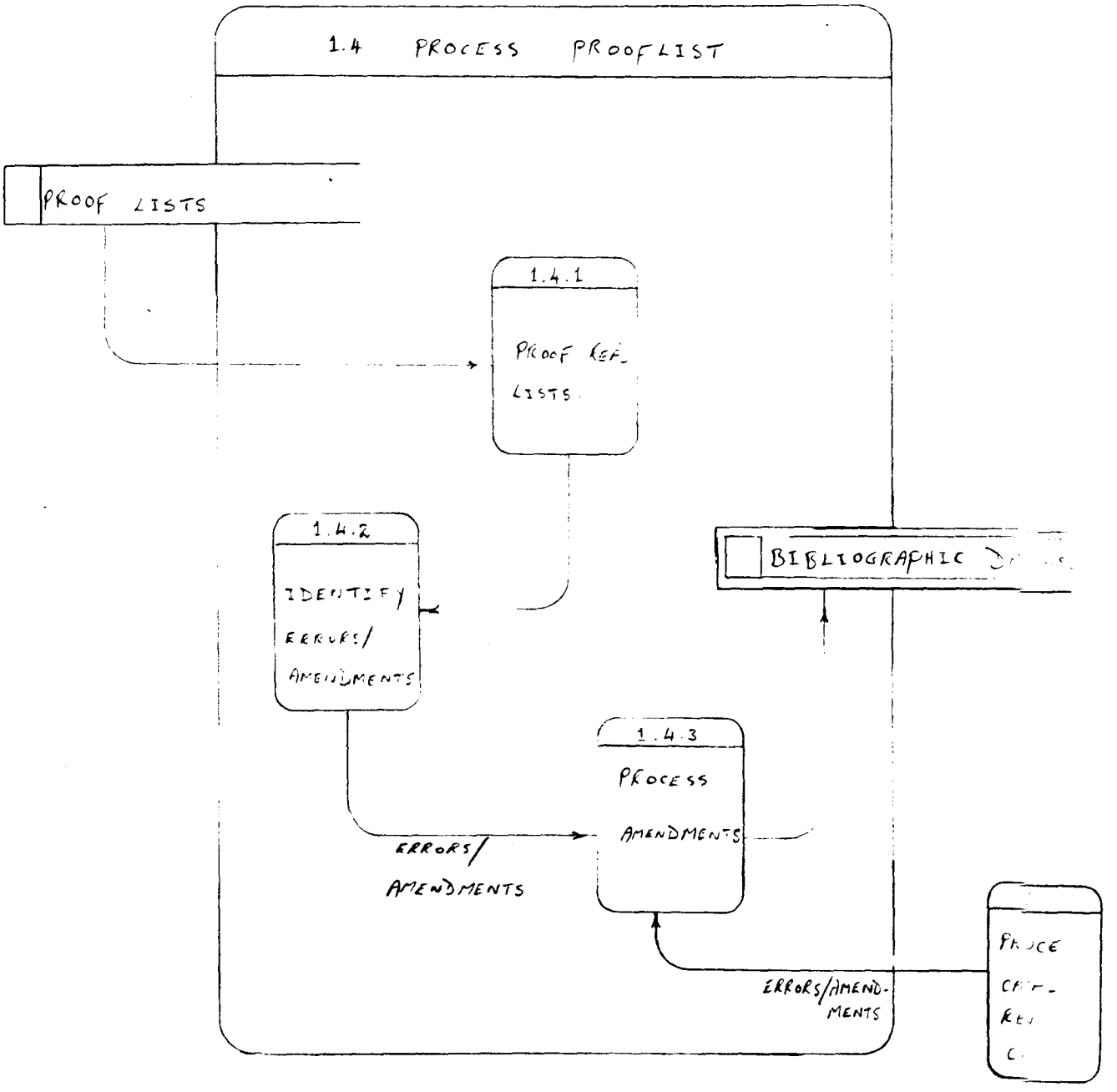
An arrow, which can be single or double ended, is used to show data in motion between external entities, processes and data stores. The arrowhead shows the direction of the flow.

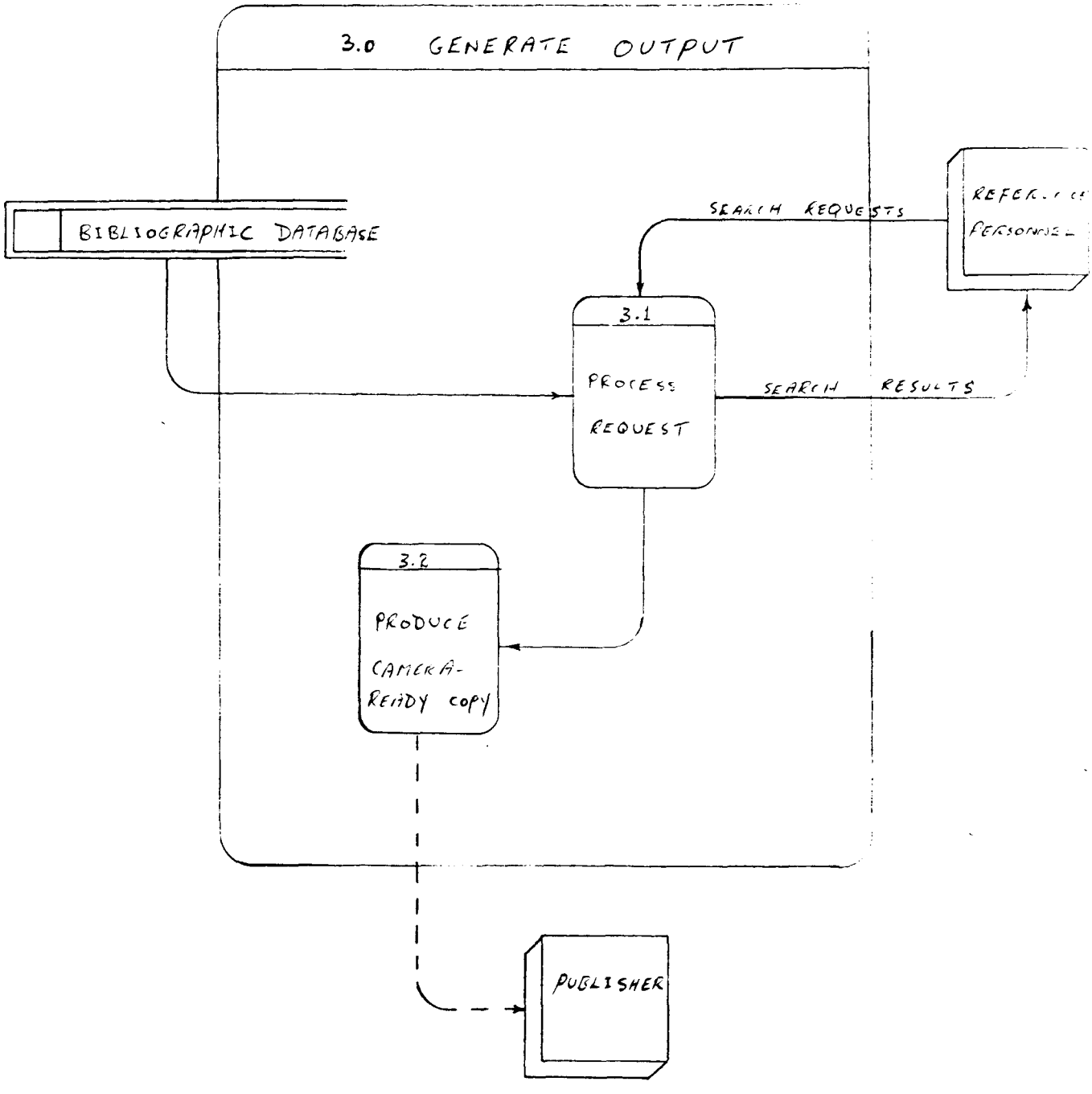
Data Flow (variation):

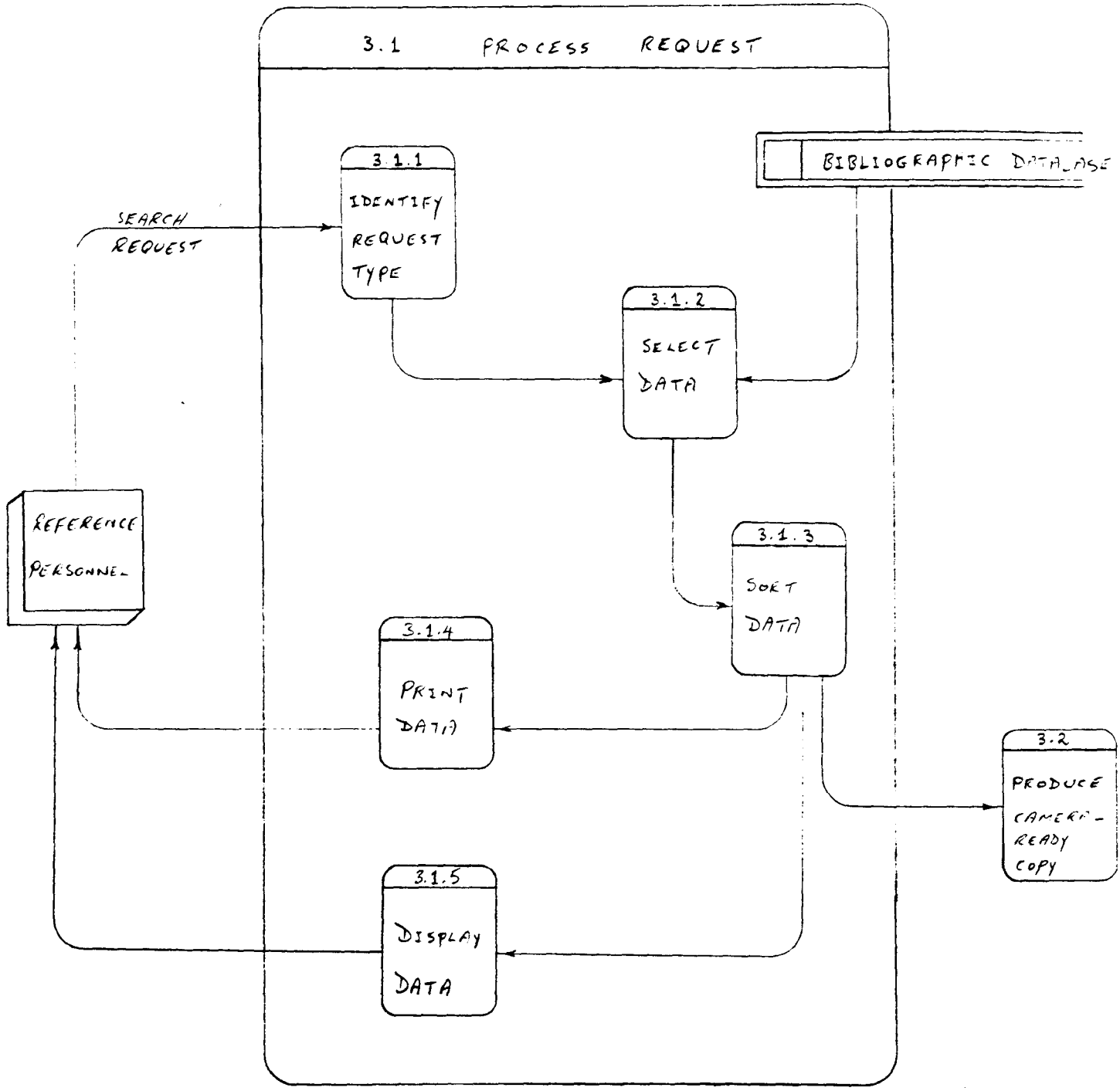
A broken arrow is used to represent data flows to and from processes/ entities/data stores which are not strictly part of the system under study. These data flows are included for clarity and completeness.

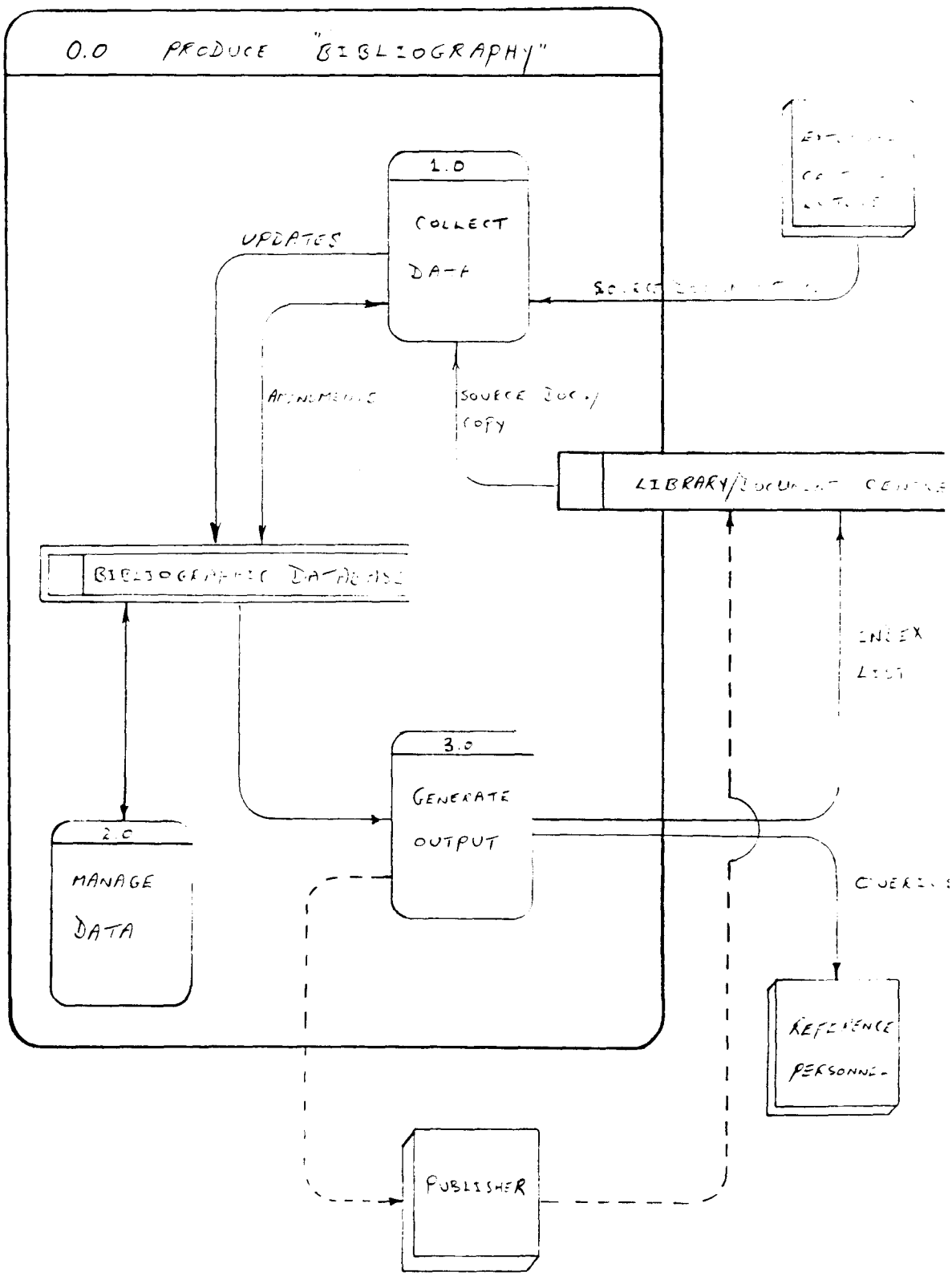

1.1 PREPARE WORKSHEET
LIBRARY/DOCUMENT CENTRE
1.1.1
DETERMINE SCOPE
SOURCE DOCUMENT/COPY
SOURCE DOCUMENT/COPY
EXTERNAL CONTRIBUTORS
SOURCE DOCUMENTS WITHIN SCOPE
1.1.3
ANALYSE WORK-SHEETS
1.1.2
CHECK FOR DUPLICATES
WORKSHEETS
1.1.4
WRITE-OUT/AMEND WORKSHEET
1.1.5
ASSIGN DESCRIPTORS
REFERENCE DATA
WORK SHEETS
COMPLETED WORKSHEETS

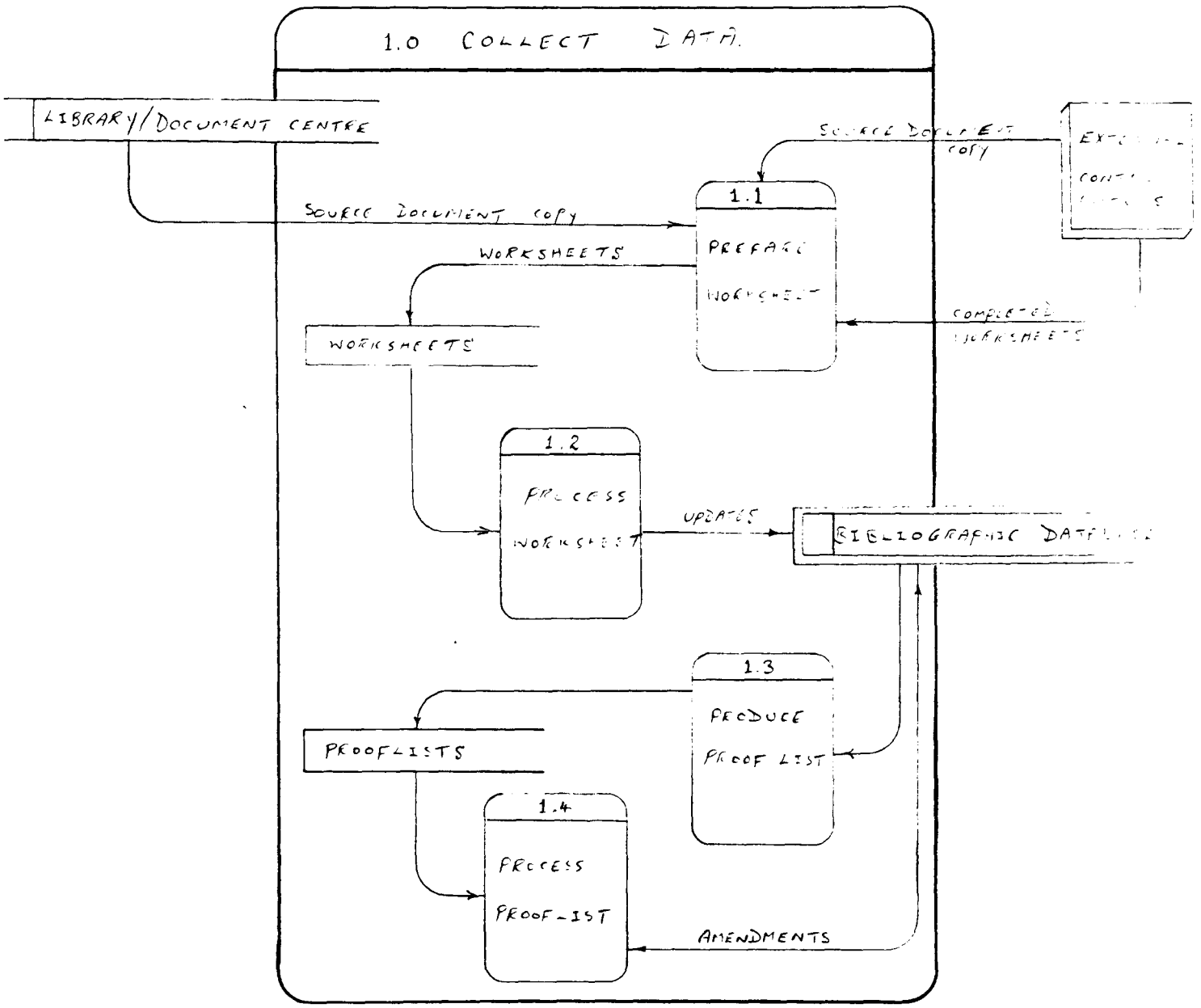

1.3 PRODUCE PROOFLIST
BIBLIOGRAPHIC DATA BASE
1.3.1
SELECT
DATA
1.3.2
SORT
DATA
1.3.3
PRINT
PROOFLIST
PROOFLISTS


1.4 PROCESS PROOFLIST
PROOF LISTS
1.4.1
PROOF REF. LISTS
1.4.2
IDENTIFY ERRORS/ AMENDMENTS
BIBLIOGRAPHIC
1.4.3
PROCESS AMENDMENTS
ERRORS/ AMENDMENTS
ERRORS/AMEND-MENTS


3.0 GENERATE OUTPUT
BIBLIOGRAPHIC DATABASE
SEARCH REQUESTS
REFER.
PERSONNEL
3.1
PROCESS REQUEST
SEARCH RESULTS
3.2
PRODUCE CAMERA-READY COPY
PUBLISHER


3.1 PROCESS REQUEST
3.1.1
IDENTIFY REQUEST TYPE
BIBLIOGRAPHIC DATA_BASE
SEARCH REQUEST
3.1.2
SELECT DATA
REFERENCE PERSONNEL
3.1.3
SORT DATA
3.1.4
PRINT DATA
3.2
PRODUCE CAMERA-READY COPY
3.1.5
DISPLAY DATA


0.0 PRODUCE "BIBLIOGRAPHY"
1.0
COLLECT
DATA
UPDATES
ARRANGEMENTS
SOURCE DOC./
COPY
LIBRARY/
BIBLIOGRAPHIC DATABASE
INDEX
LIST
3.0
GENERATE
OUTPUT
2.0
MANAGE
DATA
QUERIES
REFERENCE
PERSONNEL
PUBLISHER


1.0 COLLECT DATA
LIBRARY/DOCUMENT CENTRE
SOURCE DOCUMENT copy
SOURCE DOCUMENT copy
1.1
PREFACE WORKSHEET
WORKSHEETS
WORKSHEETS
COMPLETED WORKSHEETS
1.2
PROCESS WORKSHEET
UPDATES
BIBLIOGRAPHIC DATA
1.3
PRODUCE PROOF LIST
PROOFLISTS
1.4
PROCESS PROOF-LIST
AMENDMENTS

APPENDIX B

Database Types and Characteristics

(i) File Management Systems (FMS)

This category of software is strictly speaking outside the scope of this study, but certain software producers have marketed their FMS products under the broad category of database management systems. The development of FMS software has been an evolutionary process originating in the need for productivity tools during commercial software development. Early versions of FMS software were merely a set of independent software routines designed to interface with user programs. More recently however, user interfaces that access these stand-alone software routines have been developed, thus making them accessible to non-programmers. File management systems are adequate for applications where a single file (e.g. mailing list) is required. This type of software tends to be relatively inflexible and limited in its application, primarily due to its evolutionary path. In addition, FMS software tends to focus on the physical way in which data is stored and retrieved rather than presenting a logical view to the users. Without a clear distinction between the physical and logical view of the data, once the "database" has been created using the FMS functions the structure of the database is frozen, and to change the way in which the data is stored and retrieved, requires a re-creation of the physical structure. In many cases this is a major undertaking as the data already stored on the database will need to be reformatted to correspond to the new structure. In applications where the logical data structures are reasonably static and multiple file accessing is not a prerequisite, an FMS may be quite adequate.

(ii) Hierarchical/Network Database Management Systems (H/NDBMS)

The hierarchical and network database management systems are functionally quite similar and have, therefore, been grouped together. The H/NDBMS represents a substantial departure from the FMS approach, in that the way in which data is physically stored and manipulated is quite distinct from the logical view of the data as seen by the user of the database. The reason for making this distinction is that, more often than not, the users of a database are not interested in the way in which the data is handled internally. The purists go further in suggesting that the internal structure and data-handling mechanisms must be transparent to the user of the database, so as to maintain the focus of attention on the data and not on the way in which it is manipulated. This concept is an extension of the database concept of maintaining the independence of data from programs manipulating the data.

The origins of H/NDBMS are firmly rooted in set theory, and without wishing to digress, the basic concepts can best be explained using an example. The author of a book on "Water Contamination in Upper Volta" is a member of the set of authors who write on water contamination. The author is also a member of the set of authors who write on topics relevant to Upper Volta. We can go a step further by including Upper Volta in the set of regions that suffer from water contamination. It should be apparent that a data hierarchy is emerging, and an H/NDBMS enables a user to define the relationships and hierarchy between the various datasets so as to facilitate data retrieval.

The definition of the data sets and their interrelationships is achieved using a data definition language (DDL). The DDL also performs the translation of the logical structure into the physical structure. The physical structure of a H/NDBMS is maintained using a complex set of files, chains, pointers and linked lists, and it is these structures that are responsible for the major overheads associated with this type of database software.

Hierarchical and Network database management systems, as mentioned earlier, are similar in that a HDBMS supports one to many relations whilst a NDBMS supports many to many relations, and therefore a HDBMS can be thought of as being a special case of a NDBMS. They represent the most common type of database structure implemented on mini and mainframe computers. They have tended, however, to be much less popular on microcomputers. The primary reason for this trend has been the overheads associated with establishing and maintaining the complex internal pointers and data chains that are necessary for the physical implementation of a H/NDBMS. In addition, H/NDBMS software suffer from the same structural rigidity that is associated with FMS software. Although in the case of H/NDBMS software the level of sophistication is considerably more, the way in which the database is to be ultimately used has to be carefully planned prior to defining the database structure. The pre-planning is essential, since the structure of the database determines the way in which data can be retrieved. It is, therefore, apparent that although the physical way in which data is stored and manipulated in an H/NDBMS is quite distinct from the logical view, it is not, however, independent. This lack of independence dictates that changing the logical view of the data is in general achievable only by changing the physical structure. This apparent drawback is, on the whole, not as significant as it may first appear, particularly in the case of applications where the logical data structure and

retrieval criteria are well defined. In addition, as the capabilities of microcomputers increase, there is likely to be a corresponding increase in the sophistication of H/NDBMS modification utilities.

(iii) Relational Database Management Systems (RDBMS)

This type of software has become increasingly popular on microcomputers, and its popularity stems from the simplicity of its physical structure. An entire database is viewed as one or more two-dimensional tables that are implemented as a set of mutually independent "flat" files. The logical view of the data is controlled using "joins" and "projections". The database join enables one or more files to be logically linked together, thus presenting a new view of the data. Similarly, the view of the data can be further modified using projections, which essentially mask unwanted data whilst leaving the desired data intact. The physical database structure can be modified, by simply adding or subtracting files without affecting the entire structure. The changes to the logical view leave the underlying physical structure unchanged, and therefore RDBMS software are particularly attractive in applications where the final database structure is likely to be very different to the structure at the outset, e.g. in modelling and prototyping.

The price for simplicity in the physical structure is paid for in the sophisticated software required to manipulate the data. The run-time overheads associated with RDBMS software is, on the whole, less significant in a single user microcomputer environment than in a multi-user multi-tasking environment where there is a heavy and often conflicting demand on a variety of machine resources. Whether H/NDBMS software gain in popularity, as the capabilities of microcomputers increase, is yet to be seen. In addition, as microcomputer hardware advances, the demand made on the hardware by both system and application software is likely to increase. Whether this increased load will outstrip the hardware capabilities and, therefore, make RDBMS software less attractive, is an open question. In order to fully exploit the potential of relational database technology, the trend of implementing software features in hardware has been gaining momentum. Such firmware developments and database machines, in the mini and mainframe environment, are likely to have a profound influence on microcomputer DBMS software development.

APPENDIX C

Fast Access File Types

(i) Direct access files

Almost all DBMS software either use or provide facilities for directly accessing records. The retrieval of records is based on a relative record number or "slot" number. The relative record number can be either user definable or system definable. In any event, the ability to select a record or a range of records based on their relative position in the data file should be reported as a minimum requirement for the retrieval of bibliographic data. This type of access method tends to be fast since the operating system can easily translate the record numbers into disk addresses. The usefulness of this method is limited, however, since record numbers bear no relationship to the contents of records.

(ii) Pointer files

The accessing of data indirectly using pointer files is a logical extension of the direct access method. Pointer files contain no data as such but the record numbers of the data. With this feature it is possible to reference data in a different order by merely scanning a variety of pointer files, each maintained in a different sort sequence. The attraction of this access method is that the same data can be viewed and retrieved in different ways without physically modifying either the order or the contents of the data itself. The primary drawback of this method is the way in which the pointer files are maintained.

Most micro DBMS software create and maintain pointer files in Batch mode. The non-dynamic nature of pointer file maintenance makes this access method unsuitable for volatile data.

In the case of bibliographic systems, pointer files are of use only for sequential browsing of reasonably static data.

(iii) Index files

Index files are commonly used where searches are performed based on the contents of "key" data items (e.g. authority files). Index files are composed of key values and pointers to the data containing the keys. The retrieval process, therefore, requires scanning an index file and locating the required key and thus using the record number associated with the key as a pointer

to the data itself. It is clear that this method of indirect data retrieval has both storage and processing overheads associated with it. In addition, the following points should be considered:-

- Micro DBMS software generally provide fixed-length key indexing, however, only unique keys are supported. This limitation is relevant to bibliographic applications, since data retrieval in such applications is rarely via a unique key.

- The support of at least one alternate key and multiple index files should be mandatory.

- Index files are generally created in batch. The ability to maintain index files online whilst records are added or modified is desirable.

- The primary objective of supporting index files is to be able to retrieve specific records by key value, but in addition it should be possible to browse records in the vicinity of a selected record. This type of retrieval falls under the general topic of Indexed/keyed sequential access method which is desirable only if duplicate keys are also supported.

- Commercial micro DBMS software packages tend to define a key field as a fixed number of alphanumeric characters. The maximum allowable length of key fields varies between DBMS software products, but virtually none provide a direct capability to fragment one database field into multiple and potentially duplicate keys. This latter requirement is common in bibliographic applications where word processing is required.

(iv) Binary tree access methods

The Binary tree represents a data structure that provides fast access to data whilst minimizing data redundancy. Tree structures are well documented and are popular in applications where fast access is of paramount importance.

The building blocks of B-tree structures are referred to as leaves. Each leaf or node is made up of a data portion (containing keys, record numbers, etc.) and links to subsequent leaves in the tree. The complexity and precise structure of each

leaf tends to vary from one product to the next. However, a common feature is that each leaf has a forward link (pointing to a lower level leaf with key values greater than its own key), and also a backward link (pointing to a lower level leaf with key values less than its own key). It should be apparent that in order to retrieve a record by specifying a key value the key has to be compared with the key contained in each leaf. Depending on whether the key value is greater or less than the required value, either the forward or backward link is used to locate the next leaf. This process continues until either the leaf containing the key is found or until the tree has been completely traversed, indicating that the key is either invalid or non-existent. In the event of locating the required leaf, the record numbers associated with the leaf are used to directly retrieve the actual data. One of the main factors governing the speed of data retrieval is the number of leaves that have to be located for key comparisons. Software producers have adopted a variety of strategies, many of which tend to be proprietary information. The two most popular strategies are as follows:-

- More than one key (and corresponding record numbers) may be associated with a given leaf, and therefore more than one key comparison can be performed on the same leaf.

- When new keys are added to or old keys deleted from the tree, the software rearranges the structure (as and when necessary) to ensure that the tree is "balanced". A balanced tree minimizes the number of leaves that have to be located and tested. Some DBMS software perform structural balancing and the physical deletion of records only in batch mode. Such offline maintenance functions on B-tree files must be taken into account when evaluating the performance of this type of data retrieval.

Sequential retrieval of data (either in ascending or descending order) is relatively simple, and achievable merely by traversing the tree structure using either forward or backward links, and starting at either the top or bottom of the tree. The desirability of both direct and sequential access to data (without the use of stand-alone sort utilities) is an obvious attraction with regard to bibliographic applications.

Features for "DEVSIS-like" applications			Feature or characteristic provided by dBase II	Comments/ References
Required	Desirable	Optional		
	Menu driven		Available using command language statements, e.g. @, DO, SAY, CASE, etc.	
		Program driven	N/A	Commands not usable by external programs
Command driven			Wide range of commands available.	
	Help subsystem		Not available.	
User friendliness			Meaningful command names and error messages.	See section 2.5
Database structure definition			Limited database definition available through the use of "CREATE" command. Options of "DISPLAY" command (e.g. "STRUCTURE") provide information on database.	See section 3.1
File definition/ creation			Physical files generated by "CREATE" command.	
Record level definition			Records are numbered automatically. Only fixed-length records are supported.	See section 3.3
Field level definition			Field name, type and length specified in "CREATE". Repeatable and subfielded fields not supported. Field level security and editing not supported. Fast access file specification via "INDEX" command.	See section 3.4
	Database modification		Database restructuring supported by "COPY" and "MODIFY STRUCTURE" commands.	
		Data dictionary	Supported only in name	See section 3.6

Features for "DEVSIS-like" applications			Feature or characteristic provided by dBase II	Comments/ References
Required	Desirable	Optional		
Online data entry			Available through "APPEND", "CREATE" and "INSERT" commands. Updates performed in real time.	User definable input screens also supported
	Offline data creation		All data files can be created by external programs. Database "loading" supported using "COPY" command.	See section 4.2
Online data modification			Available through the use of "EDIT", "REPLACE", "CHANGE" and "BROWSE".	
	Global data modification		Available only indirectly, using command language (e.g. DO WHILE).	
Data deletion			Supported using "DELETE", "RECALL", "PACK".	
Online searching			Supported using "FIND", "LOCATE", "DISPLAY", "LIST" and "REPORT". Free text searching also supported. Boolean searching supported. Word and term processing not available directly.	See section 6.1
Batch extraction			Available using a command file.	
	Fast access files		"INDEX" command creates B-tree fast access file.	See section 6.3
		Multiple user views	Limited support through "JOIN" command. Projection not supported, though "COPY" provides selective data extraction.	
		Arithmetic computation	Arithmetic and Boolean operations provided. Parenthesis are supported. Temporary memory variables are available for computation.	

Features for "DEVSIS-like" applications			Feature or characteristic provided by dBase II	Comments/ References
Required	Desirable	Optional		
Output creation			Supported using "DISPLAY", "LIST" and "REPORT".	
Print format definition			Supported using the command language.	
Output re-direction			"SET" and "COPY" commands provided.	
	Sort/merge		Supported using "SORT" and "INDEX" commands.	
	Database security		Limited security provided by disabling the "ESCAPE" command.	
		User passwords	Not supported.	
		File locking	Not supported.	Copying and sorting to a file in "USE" can corrupt data.
		Transaction logging/ recovery	Not supported.	
System backup			Not supported.	Performed using operating system utilities.

Features for "DEVSIS-like" applications			Feature or characteristic provided by dBase II	Comments/ References
Required	Desirable	Optional		
	Software configurability		Limited configuration supported using the "INSTALL" program.	
Disk file chaining			Not supported.	Hard disks are supported.
		Programming language interface	Not supported.	Sophisticated command language provided.
	User defined commands/ macro generation		Macro substitution supported.	
	High level procedural language subsystem		Comprehensive command language provided.	
		Multi-user capability and local area networking	Not supported.	
	File compatibility		All database files are in ASCII text file formats.	

dBase II Specifications

- Number of database records	65 535	max.
- Characters per database record (fixed-length records)	1 000	max.
- Fields per database record	32	max.
- Characters per database field (fixed-length fields)	254	max.
- Files that can be open simultaneously	16	max.
- User-definable variables	65	max.
- Key/Index file(s) accessible simultaneously	7	max.
- Characters per individual key	100	max.
- Number of databases accessible simultaneously	2	max.

Hardware Requirements

- CPU	Z80, 8080, 8085/6/8
- Minimum space requirements	48 K
- Storage requirements	One or more disk drives
- Video display terminal	24 x 80 cursor-addressed
- Hard disk	Optional
- Printer	Optional

System Software

- CP/M, MSDOS

APPENDIX E

Software Evaluation Forms

The following forms are intended to assist a potential software evaluator in assessing the desirability of a given software product.

The section references have been included as an index to the main body of text.

DBMS Features	O	D	M	Section Reference	*General Comments (Features provided directly/indirectly)
Database definition					
Database structure definition			X	3.1	
File definition/creation	X			3.2	
Record level definition				3.3	
Sort specification	X				
Record number/identifier specification		X			
Default print format specification		X			
Variable length record specification		X			
Field level definition				3.4	
Field identifier specification			X		
Field data type specification	X				
Field length specification		X			
Repeatable field specification			X		
Subfield specification		X			
Field editing specification	X				
Database modification	X			3.5	
Data dictionary	X			3.6	
Database population					
Online data entry				4.1	
Prompted entry					
(i)	X				
(ii)			X		
(iii)			X		
(iv)			X		
(v)		X			
(vi)	X				
(vii)	X				
(viii)		X			
(ix)	X				
(x)			X		
(xi)		X			
(xii)		X			
Formatted entry	X				
Offline data creation			X	4.2	

* - This column is to be filled in by the software evaluator.
O - Optional feature.
D - Desirable feature.
M - Mandatory feature.

DBMS Features (continued)	O	D	..	Section Reference	*General Comments (Features provided directly/indirectly)
Data modification					
Online data modification			X	5.1	
Global data modification		X		5.2	
Data deletion			X	5.3	
Data retrieval					
Online searching				6.1	
Free text searching		X			
Searching using a range of record IDs		X			
Word and term processing/key generation		X			
Boolean and nested search logic			X		
Back referencing		X			
Storage and retrieval of search strategy		X			
Special features	X				
Batch extraction		X		6.2	
Fast access files	X			6.3	
Arithmetic computation	X			7.0	
Output generation					
Output creation			X	8.1	
Print format definition				8.2	
Page level formatting					
(i)			X		
(ii)	X				
(iii)			X		
(iv)		X			
(v)	X				
(vi)		X			
(vii)					
Record level formatting					
(i)		X			
(ii)	X				
(iii)		X			
Field level formatting					
(i)			X		
(ii)	X				
(iii)	X				
(iv)			X		
(v)			X		
Output redirection			X	8.3	
Sort/merge			X	8.4	

DBMS Features (continued)	O	D	M	Section Reference	*General Comments (Features provided directly/indirectly)
Data integrity					
Database security		X		9.1	
User passwords	X			9.2	
File locking		X		9.3	
Transaction logging/recovery	X			9.4	
System backup	X			9.5	
Utilities and special features					
Software configurability	X			10.1	
Disk file chaining		X		10.2	
Programming language interface		X		10.3	
High-level procedural language subsystem		X		10.4	
User-defined command/macro generation		X		10.5	
Multi-user capability and local area networking	X			10.6	
File compatibility		X		10.7	
Database loading and unloading		X		10.8	