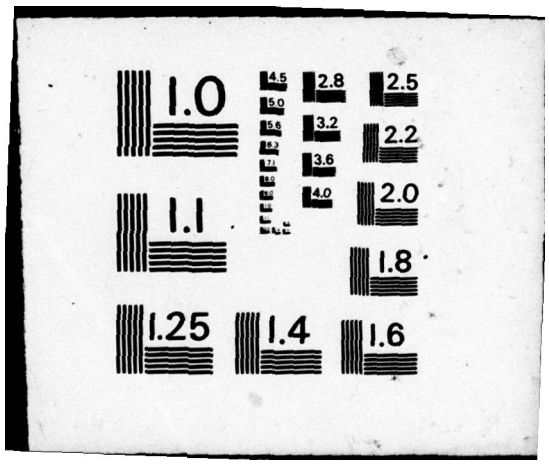
AD-A036 065 SYSTEM DEVELOPMENT CORP SANTA MONICA CALIF F/G 9/2
SOFTWARE DATA COLLECTION STUDY. VOLUME IV. DATA MANAGEMENT SYST--ETC(U)
DEC 76 M P TEMPLETON F30602-75-C-0248
UNCLASSIFIED SDC-TM-5542/004/01 RADC-TR-76-329-VOL-4 NL

1 OF 1
ADA036065

END
DATE FILMED
3-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

RADC-TR-76-329, Volume IV (of eight)
Final Technical Report
December 1976

# SOFTWARE DATA COLLECTION STUDY
## Data Management System Interface

System Development Corporation

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public including foreign nations.

This report has been reviewed and is approved for publication.

APPROVED: Richard T. Slavinski

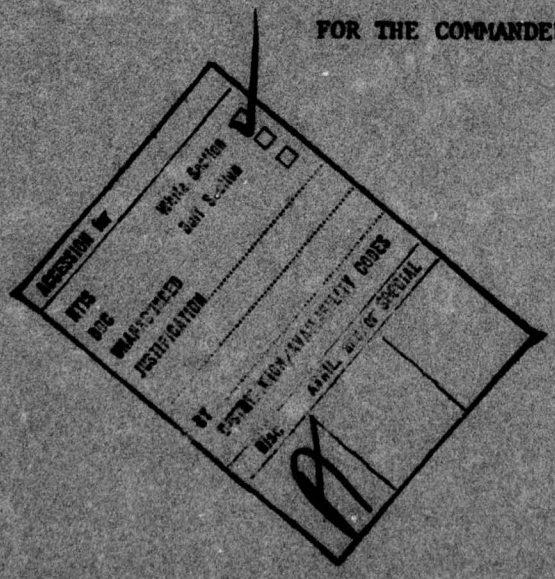
RICHARD T. SLAVINSKI
Project Engineer

APPROVED: Alan R. Barnum

ALAN R. BARNUM
Assistant Chief
Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office




Do not return this copy. Retain or destroy.

# *MISSION*
# *of*
# *Rome Air Development Center*

*RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications ($C^3$) activities, and in the $C^3$ areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.*

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER: RADC-TR-76-329, Vol-II (of eight) | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle): SOFTWARE DATA COLLECTION STUDY. Volume II. Data Management System Interface. | | 5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report, Jun 75 — Jun 76. |
| | | 6. PERFORMING ORG. REPORT NUMBER: SDC-TM-5542/004/01 |
| 7. AUTHOR(s): M. P. Templeton | | 8. CONTRACT OR GRANT NUMBER(s): F30602-75-C-0248 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: System Development Corporation, 2500 Colorado Avenue, Santa Monica CA 90406 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 63728F, 55500810, 08 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (ISIS), Griffiss AFB NY 13441 | | 12. REPORT DATE: December 1976 |
| | | 13. NUMBER OF PAGES: 72 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same, 78p. | | 15. SECURITY CLASS. (of this report): UNCLASSIFIED |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer:
Richard T. Slavinski (ISIS)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Entry
Data Collection
Data Management

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Data collected by the RADC Data Collection and Repository System may be stored as documents and/or entered into a data base. This document explores alternate data structures, data management systems, data storage devices, and data entry devices that may be used for the data base.

General recommendations are made based on the need for flexibility, security, growth potential, and ease of implementation and use. The basic recommendations

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

339 900

are:

- Data should be entered via intelligent terminals at the source.
- The data base should be close to the point of use which means some decentralization.
- Data should be stored in serial records and accessed via indexes.
- Data access should be through a data management system which separates the logical view of the data from the physical structure and protects the data from unauthorized access or change.

## Table of Contents

## Table of Contents (cont'd)

## 1.0 DATA MANAGEMENT SYSTEM INTERFACE

A data base will be needed to preserve the software development data that is collected. The design of such a data base depends upon the volume and type of data that is to be collected, the uses that will be made of the data, and the budgeted cost of the system. At the same time, the cost of storage, transmission, and retrieval influences the volume and type of data that can cost effectively be collected. Therefore, before data base designs are discussed, the alternatives for data structure, data management functions, and hardware are discussed. These building blocks will be used when suggesting alternative designs in a later section.

## 2.0 DATA STRUCTURES

Data are a representation of information. In order to be useful for processing, data must be stored in an orderly way on some device where it can be located again when it is needed. This implies that there is some physical order of the data on a device, that there is some technique for accessing the data, and that it is organized in some fashion. This section discusses the various methods by which this can be done.

A survey of physical organizations, access methods, and logical structures is made with an emphasis put on those that are candidates for use in the data collection system. It is likely that a data management system used for the data collection system will combine several of the data structures discussed.

An evaluation is made of each structure based on its flexibility, security, growth potential, and ease of implementation and use. The discussion at this point assumes some general characteristics of the data collection system but does not assume knowledge of the data items to be collected or the volume of data or the exact use of the data. The assumptions made are:

- the data base will be used for project management and research

- there will be an increasing volume of data
- there will be changes in the data items collected
- the system must be flexible enough to collect different types of data from different users and easily add users and change what is collected
- the queries will not be known in advance
- a users's raw data should not be visible to other users except anonymously
- a user can update only his own data
- the data base must be secure from malicious users
- no specific hardware has been chosen

The discussion starts with physical organization which is the format used to write the file on the device. This level is not seen by the system user and may or may not be seen by an application program. Access methods are then discussed. This is one logical step up from the physical level and is the level often seen by the application program but not the system user. Finally, the logical structures are discussed. The logical structure is a view of a data base rather than of a file. This level affects the user view of the data. Query languages are based on one of the logical views of the data base. In some cases application programs also view data at the logical level. Data management routines then map the logical view to the physical view by consulting a directory.

## 2.1 PHYSICAL ORGANIZATION

Data on a device is written in a block in one of three basic ways: serial, linked, or direct. One block may contain one logical record, several records, or part of a record. A fourth organization, the fully inverted organization, crosses the bounds of a physical organization and an access method.

### 2.1.1 Serial

Serial means that records are in physical sequence with one block following the next on the device. In order to read a file, each block is read in order. The blocks may be sorted by some field or the "sort" order may be the time sequence when the block was written.

Serial physical organization:


Serial organization is the safest from a restart standpoint. Each block is independent and, if a rewrite or write fails, it can be tried again. It is also easy to implement.

A disadvantage of serial organization is that record lengths cannot be changed and records cannot be deleted from the middle of a file or added to the middle of the file so that order based on some field can be maintained only by resorting the file. If the data base is large, sorting a file is expensive.

Serial organization uses space efficiently, grows easily, and can be used as the basis for several access methods. Serial devices such as tape must use serial physical organization. It is also frequently used to store records on direct access devices so that records may be written sequentially in the next space but then retrieved sequentially or randomly through one or more indexes. More will be said about this when access methods are discussed.

### 2.1.2 Linked List

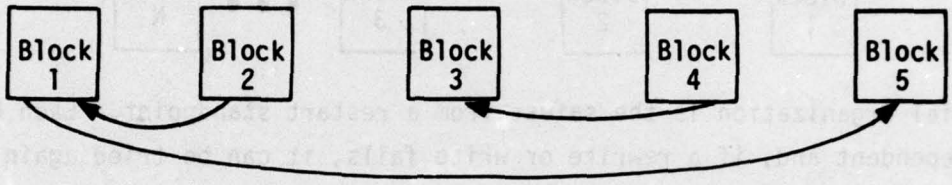
Blocks in a linked list are chained within a file or within an area of a file called a bucket or cell. Each block contains a pointer to the next block. The pointer can be a full disk address, a relative block number, or a relative track and record number. Generally a full disk address is not used because it makes the file unmoveable. The block is located either by a fixed position or through a index; the pointer for the next record is then taken from the first record.

There are many variations of linked lists. The simplest variety is a file with a chain of records ordered on some field which is called the prime key field and a chain of records that are unused:

Set pointers:

Pointer to beginning of list of used blocks - 2
Pointer to beginning of list of unused blocks - 4

Records in file:




Many more complex chain structures have been used. Pointers may point to the previous record as well as to the next record. Records within a block may be chained rather than chaining the blocks. One record or block may belong to several chains based on the values of one or more fields. Areas of the disk may be chained while blocks within the area are written sequentially.

A file with linked list physical organization has the advantage of keeping records within a file in order without having to sort the input or output. Complex networks of relationships can be represented which can support a powerful query language. However, the disadvantages are many. Logical relationships are expressed in physical structure which makes it difficult to add or change relationships without reorganizing the data base. Considerable space may be required to hold pointers. Access time can be slow because records are accessed through lists; if a chain is a long one it may take a long time to traverse a chain to reach the desired record. Access time gets worse as the data base grows and reorganizations are necessary to keep chains from spanning too many cyclinders.

Linked lists pose a recovery problem when the system malfunctions. When a record is written at least one other record must also be rewritten to update a pointer. With some organizations, forward and backward pointers must be rewritten. If the two or three writes are not successful, the chain is broken

and all records down the chain are lost. This is not a severe problem if the data base can be dumped periodically in full or part and restored when a problem occurs.

Because of the many disadvantages, the linked list should not be used for storage of records in the data collection data base. It could, however, be used in conjunction with other physical structures. Mention will be made of various places where the linked list structure is useful.

### 2.1.3 Direct

Blocks that are stored in a direct file are stored in an address based on the value of the prime key or identification field. The key is "hashed" by some technique to get to an address. There are many ways to hash a key depending upon known characteristics of the key. The simplest method is to have one bucket for each possible key. This only works if the keys are unique. When these criteria are not satisfied, one of many hashing techniques may be used [16, 31]. The best general purpose technique is to divide the key by the prime number (or almost prime number) closest to the available number of buckets or addresses. The remainder becomes the bucket number in which the record belongs. A bucket may be a single block holding a single record, a track, a cylinder, or some other fixed size area. If the bucket is large, the bucket itself may need to be organized in some way such as with a linked list.

The chief problem with hashing techniques is that there are nearly always overflow problems; more keys belong in a single bucket than will fit. Overflow blocks may be stored in separate overflow buckets or in adjacent buckets or in low useage buckets. The overflow blocks may be located by linked lists from the home bucket or by searches of overflow buckets. The handling of overflow is the main distinction between the various algorithmic access techniques that are based on direct storage.

Direct organization has the advantage of access to a record, given its prime key, with as few as one disk read. However, it may require more accesses than a sorted sequential organization requires if records are being accessed by a secondary key. Direct storage also consumes extra space because the fuller the file gets the more chance that a new key will hash to a full bucket. Overflow decreases the efficiency of the file. For an average type of key that neither maps directly to an address nor is extremely skewed, the file must contain at least 5% empty space. If records are large (over 100 bytes) a loading factor of 80% is desirable. If records are small (100 bytes) a loading factor of 85-95% is desirable. In either case, a loading factor over 95% grinds the system to a halt chasing long overflow chains [31, 20]. It is difficult to control loading factors in a growing file; for this reason direct organization is best used for a static system.

### 2.1.4 Fully Inverted

In a fully inverted file, records do not exist as a unit of information. When a new entry is created, each attribute value is stored in a separate index.

This organization is useful and fast for answering queries of the type, "how many entries have attribute X", but it does not perform well for a query of the type, "list all attributes of entry X". It is also difficult and slow to delete or change an entry because the pieces of it must be located in all the indices and deleted if the record is being deleted or changed and refiled if the field is being changed. For these reasons, this would not be a good primary organization for the data collection system.

## 2.2 ACCESS METHODS

The access method is the interface between a program and the physical file. The discussion here covers four access methods that may be used singly or in combination to access files organized in the four physical formats: serial, linked list, direct, and fully inverted. All access methods cannot be used for all file organizations. The chart lists physical organizations and possible access methods.

**Access Method**

| Organization | Sequential | Chained | Algorithmic | Indexed |
|---|---|---|---|---|
| Serial | X | | | X |
| Linked List | P | X | | P |
| Direct | P | | X | P |
| Fully Inverted | | | | X |

X = generally used

P = possible in some cases

A data management system may be built from a single access method or many access methods may be used together. For example, the data records in a file may be written serially and accessed sequentially in order of receipt of the record. There may also be an index to the sequential file which uses direct organization and uses linked lists for overflow records.

All access methods provide the basic functions of checking for I/O errors and calculating the hardware address of the record given a key (which ranges from a relative disk address to the value of a field). Some do much more than this.

### 2.2.1 Sequential

The sequential access method is the simplest method. The first block in a file is located by the start of device, in the case of a nonlabeled tape or card reader, or by the first address for a file in the case of a shared device such as a disk. The block is transferred to a core buffer where it is unblocked if necessary. The next read then causes the next record in the block to be passed to the program or a new block to be retrieved from the device. The next block retrieved is the next block after the interrecord gap or the first block on the next track.

The partitioned access method is a variation on the sequential access method. Both are used to access files built in serial physical sequence but the partitioned access method requires direct access storage. A directory is built which points to sections of the file by name and address. The accessing program locates the desired member and then reads the member as though it were a regular sequential file.

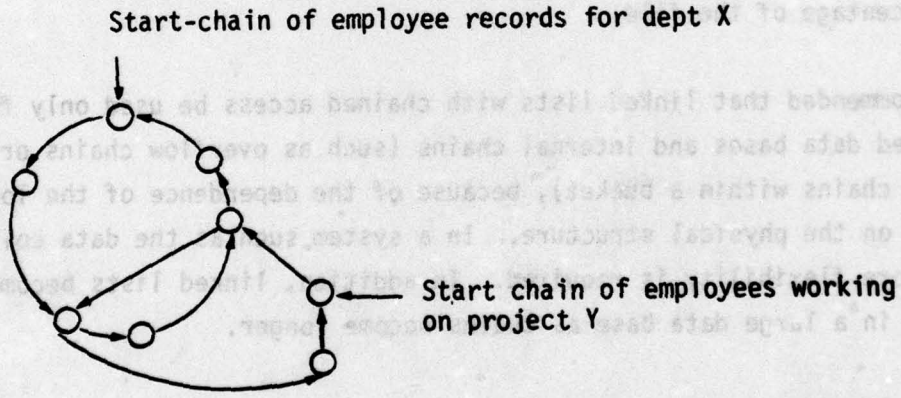
The sequential access method is the only one available for serial devices such as tape. It is also the fastest way to access a data file when over 8% (varies with blocking factor) of the file is to be accessed and the order of access is not important. This would be true when answering queries of the type: "list all entries where X is true" or "how many have attributes X and Y".

This access method perhaps offers less security than others because the user program does have access to the entire file. It is not necessary to know the key of a record in order to retrieve it. It is possible, though, to block access to the entire file by a password or by removing the tape or disk pack from the system. It is also possible to provide a security screen which selects only the records that a user is allowed to see. The user asks for the "next" record which may not be the next physical record but the next record that the user is authorized to see.

Sequential access presents records in the order of storage which may not be acceptable for reporting. On the other hand, sorting reports consisting of selected records can be much cheaper than maintaining complex structures to retrieve data in every order used by a report.

### 2.2.2 Chained

Chained access is used to access a file that has a linked list organization. The start of the chain pointer is located from a fixed address or from an index record. The pointer is then converted to a full disk address if the pointer is not already an address. Every record in the chain is then searched to find the desired record or records. Chains may be followed to a point where a new chain is picked up and followed. An example will clarify this:



In this example, there are six employees in department X, two of whom work on project Y. Two other employees outside of the department also work on project Y. The department chain can be used to list all employees in a department. The project chain can be used to list all employees working on a project. Both chains could be used to list all employees of the department and their coworkers.

Chained access is good for getting lists of records with certain attributes if the attributes in the query were known in advance and were built into the structure. It is difficult to get lists based on attributes that are not the basis for chains because it requires a pass of the entire file. Even if a chain exists which includes all records in a file, the complete pass may be more expensive than passing a serial database if it is necessary to skip around the file. It may be possible to treat the file as a serial file for sequential processing if unused records can be recognized and if they are a small percentage of the file.

It is recommended that linked lists with chained access be used only for specialized data bases and internal chains (such as overflow chains or index chains or chains within a bucket), because of the dependence of the logical structure on the physical structure.. In a system such as the data collection system, more flexibility is required. In addition, linked lists become expensive in a large data base as chains become longer.

### 2.2.3 Algorithmic

Algorithmic access is used to access files with direct organization. The same techniques that were used to store records are used to access the file. The desired key is hashed using the same hashing algorithm that was used to store the record. This yields the address of a bucket. When the record is not located in the bucket, the overflow areas are searched.

Algorithmic access is useful when fast access to a large file is required and records are being retrieved by prime key. Access requests should be of the type: "print attributes for record X". This access technique is not good for requests of the type: "How many entries have attributes X and Y" or "list all entries where X is true" or "list all entries with key X1 thru X10." Because of the unknown nature of the queries directed to the data collection data base, this access method is not practical.

Algorithmic access is more secure than sequential access because the user does not have access to all records; the prime key must be known in order to access a record.

### 2.2.4 Indexed

Indexing of one sort or another is the basis for most data management systems. The file may be organized in any physical order: serial, linked list, direct or fully inverted. As a record is added, deleted, or changed an entry is made or removed from one or more indices. The index contains one or more values of one or more attributes and the addresses of records associated with the values. The address may be the prime key or a relative disk address or an absolute disk address or no address in the case of the fully inverted file.

#### 2.2.4.1 Choosing Keys

A key is an attribute that will be indexed and used for access. In the case of the fully inverted file every attribute is indexed. With other files, one key is selected as the prime key for a file. This key is the field or fields whose value is hashed to locate the address if the file has a direct organization, or it is the key that is used to index all records in the file. A field is chosen to be the prime key if it is frequently used to access the data and the values of the field are unique or at least there are not many records with a single value. With some access methods, such as IBM's ISAM, the prime key must be unique so if it isn't it must be appended with a field which will make it unique.

Indexes may also be built for other fields which are called secondary key fields. With some physical organizations, the fact that one key is called "prime" and the other is called "secondary" is purely arbitrary. With direct organization or when only the prime key index contains a disk address, there is a difference between a secondary and primary key because the secondary key is used to locate the primary key which is used to locate the record.

The cost of a secondary index must be weighed against the value of the index. The value is the savings in selecting records via the index rather than passing the entire file when the candidate key is used in qualification. The cost is the cost of building and maintaining the index. The more updating that is done on a field, the higher this cost. When the queries to be made on a file are known and the create, update, and delete activity is known, it is possible to weigh the cost of the index against the value using curves developed by a simulation model [31]. When the parameters are not known, an educated guess is all that can be done. The system should be flexible enough to add a secondary index after experience is gained with a file.

Until this experience is gained, the guess can be based on knowledge about the number of different values that may be taken by the candidate key field and the distribution of the values which determine the selection ratio (the likelihood that the field is used in qualification), the cost of a sequential file search, and the probability that the field will change. The selection ratio will be high if there are few different values; unless there are many different values it probably will not pay to index the field because if 8% or more of the records are selected, it is as economical to pass the file sequentially. (The 8% figure is approximate and depends upon the blocking factor and number of levels of index). For example, a sex field is not a good field for an index if 50% are men and 50% are women. It might be a good field for an index of only women if there are 2% women in the file. In the later case, no index would be built for men.

A small file is cheap to pass so the extra program overhead associated with an index would not pay. On the other hand, if there are a million records in a file the cost of passing them may be prohibitive and therefore every field that is ever used in qualification may be indexed.

The probability that the field will change is the third variable to consider when deciding whether or not to build an index for a field. It will require at least one read and write to update an index block when a record is created, when the indexed field is changed, or when the record is deleted. If the update level is high, the extra accesses required to locate a record through a non-indexed field may be worth it.

A secondary index may be built using a combination of fields for a key if they are generally used together in qualification [18]. In some cases, a field may seldom be used alone or it may not be used frequently enough to qualify as an index field. It then is a candiate for a combined index. Two or three fields may be combined as one index value. The field with the highest probability of being present in the query qualification comes first, followed by fields in order of decreasing probability of being present. When the probability is close to equal, an index in each order is necessary unless an index already exists on the first field. The combined index is then examined with the same rigor as a single field index; the selection ratio, the cost of a sequential file search, and the probability that the field will change are the criteria used when determining whether a single field should be indexed.

An additional variable in the combined index decision is the size of the index. The value field in a combined index is larger than the value entry of one of the fields alone. Therefore, each index entry is larger and the number of unique values is larger which will increase the size of the index and the index access time. The probability of change may also be higher because the probability that the combined field will change may be greater than the probability that the component fields will change.

The type of logic that must be followed can be illustrated with an example. Assume that field A is indexed on its own merits and that it is often used with field B, which is not indexed alone. A combined index will slow down access to field A alone but will speed up access to the combination. It also increases the probability of a change in the index entry because the probability that only B changes is added to the probability that A changes. If B has few different values and therefore a selection ratio over 25% it will not pay to use a combined index. It would be better to access all records in index A and bring them to see if they qualify on field B.

Note that a selection ratio of 25% is used in the examples. An 8% selection ratio was used when discussing whether the file should be passed sequentially. The difference is caused by the overhead involved in reading a record randomly through an index because chances that the record will be contained in a block in core are less and multiple reads must be done to access index blocks.

2.2.4.2 Index Organization

The basic physical organizations of serial, linked list, or direct may all be used for an index. A direct index may be used when a direct file is indicated; i.e., there is a primary key that is used most of the time, fast access is required, and the size of the file is fairly stable. By making the index direct rather than the file, DASD (direct access storage device) space may be saved because a given percentage of overflow space on an index means fewer tracks since an index entry is generally smaller than the corresponding record. Files with direct organization generally must have fixed length records but, by making the index direct rather than the file, the file can contain variable length records. The disadvantages of a direct index over direct data organization are that access now requires two reads instead of one, the index must be changed when the key changes, and DASD space is required for an index.

A serial index is more common. There are several variations on this organization. There could be a list with the value for each key of each record followed by the record address. This list could be sorted or unsorted. If it is sorted, a binary search can be done on the index. The index could also be organized with a list of each possible value or each represented value followed by a count of the number of records containing the value and the addresses of all the records. This second method has an advantage over the one to one list in that less space is consumed when there are many duplicate values and "how many" type requests can be easily determined from the index; the disadvantage is that insertion is more difficult.

An index may be dense or sparse. A dense index contains every value and address. A sparse index contains selected entries only. It may point to the highest key in a bucket or to the highest key on a track, if records are sorted, or it may point to another index which has every value and address. This later technique is referred to as levels of index. When every address and value of a large file is contained in an index, the index search may require a large number of accesses. By indexing the index records, the search time can be cut down.

The values that are contained in an index may be compressed by one of several techniques [20]. This saves space but requires some CPU time to reconstruct values and a binary search cannot be done on the index. It may pay if the index is large and infrequently accessed.

#### 2.2.4.3 Index Searching Techniques

The index may be searched to locate the address of records which may qualify or it may be used to satisfy the query. If a query is of the type "how many records are there with attributes X and Y" and both X and Y are indexed, the records need not be retrieved because the query can be satsified from the index. However, if X is indexed and Y is not, each record with attribute X would have to be retrieved to check whether it also has attribute Y.

There are several ways to search an index [20]. Scanning the index from start to finish is slow on a large index but is effective for a small index or one with many levels, each a small index. A binary search is effective if the index is in sort order and contains fixed length entries and can be contained in core. In either case, as much of the index as possible should be core resident. If the index is too large to be completely core resident, an area can be reserved for the parts of the index that are in use. When new index blocks are needed the new blocks can replace blocks that have low activity or that were not recently accessed.

#### 2.2.4.4 Security

Access through an index provides a level of security that does not exist with sequential access. Different users may be given different sets of records which are accessed through an index. For instance, the originator of a record could be a field within a record. If an index is constructed for each originator code, access could be blocked to records not filed in the index. This is effective only if all access is done through the same data management system.

## 2.3 LOGICAL STRUCTURES

The logical structure is the view of the data that the user sees. The logical structure is a view of a data base rather than a file. In some cases programs see only the logical view and in others the program must do the mapping from the logical, user view, to the access method.

The logical structure may be hard coded into a program but it is generally defined via a data definition language. The data definition language ranges from those that describe only the physical structure to those that cover the aspects recommended by SHARE [32]: field specification, set relationships, key, access authorization, required fields, validation rules, and encoding rules.

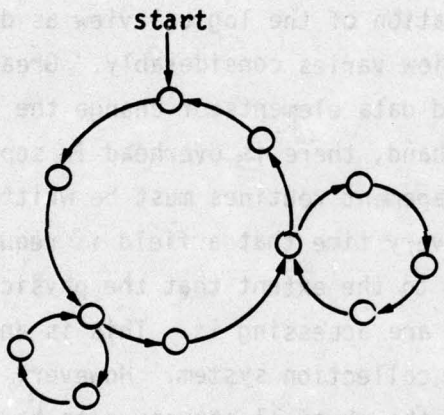
Data definitions can also vary from user to user. The CODASYL committee uses the terms "schema" and "subschema" [5]. The schema describes the data base and the subschema is the view of one user. Each user may have a different subschema depending upon his need to know and the way in which he uses the data. The subschemas are controlled with privacy "locks" and "keys". Unless a user has the "key" to the subschema, he cannot look at the data base. This protects the data base from a user who accesses it through the data management system. It does not protect the physical data base from access by a specially written program unless the data as stored is scrambled or encoded.

The amount of desirable separation of the logical view as defined in the various subschemas and the physical view varies considerably. Greater separation means that it is easier to add data elements or change the relationships between files. On the other hand, there is overhead in separating the logical and physical views. Data management routines must be written to locate desired fields and must be executed every time that a field is requested. There are advocates of total separation to the extent that the physical data base could be reorganized while programs are accessing it. This is an expensive luxury for a system such as the data collection system. However, it would be desirable for the logical subschema and the physical structure to be separated to the extent that new data items can be added to records without recompiling programs. Data items should also be convertable from their internal form to the desired external form. This may mean a change of representation (i.e., character external, binary internal), decoding (i.e., full descriptions external, code internal), or expansion (i.e., compressed format with strings of repeated characters dropped for storage). More than one logical view of the data base should also exist to prevent every user from seeing everything, or, to put it more positively, to accommodate the view to the user.

The comments about the separation of the logical and physical view apply to all logical views. Logical views are then divided up into three basic classes or ways of looking at a data base: the network view, the hierarchical view, and the relational view. Each of these will be discussed separately.

### 2.3.1 Network

The network view is generally associated with the linked list physical organization. The user views the data base as a series of chains which may be one or two way and which may intersect. The data base view is:




The nodes are logical records which may be of one or more types and may reside in one or more different files.

The network view is best illustrated by describing PROTEUS, a network data management system owned by SDC.

In the PROTEUS system, all records reside in one physical file which is logically partitioned into element groups. Records in one element group share a record format and are linked together in order of their unique key for sequential access. One or more records from one element group may also be chained to a

record in a different element group to form a subelement group. Subelement groups always cross element group boundaries. No one record in the "owned" element group may belong to more than one record in the "owner" element group within the same subelement group relationship. However, multiple subelement group relationships may be set up between the element groups.

Chains of the form shown in Figure 2-1 may be established. Note that an element group is shown approximately in order of the keys. It is maintained in order of keys by periodic reorganization for efficiency. However, it will function out of order. Also note that pointers are shown as forward pointers only for clarity. Backward pointers are also maintained in the actual PROTEUS data base.

The user program may navigate through the data base by starting with any element group and then retrieving a record by the unique ID or sequential position. Any subelement group chain may then be followed forward or backward by locating the next record in the chain or the 'nth record from the current position. Once a record from a subelement group is read, it may be used as the starting point for following the chain through that element group or through another subelement group. The only restriction is that the same record may not be retrieved more than once to prevent the program from getting into an endless loop.

The network view as illustrated by PROTEUS is very powerful for linking records in multiple ways. The chief problem is that the user program must be able to keep track of where it is and where it is going which can make it a very complex program. Deleting records is also a problem. A record must be removed from every chain in which it is a member which requires reading the previous and following members to remove the links. If a record is an owner, a decision must be made about the disposition of the entire subelement group.

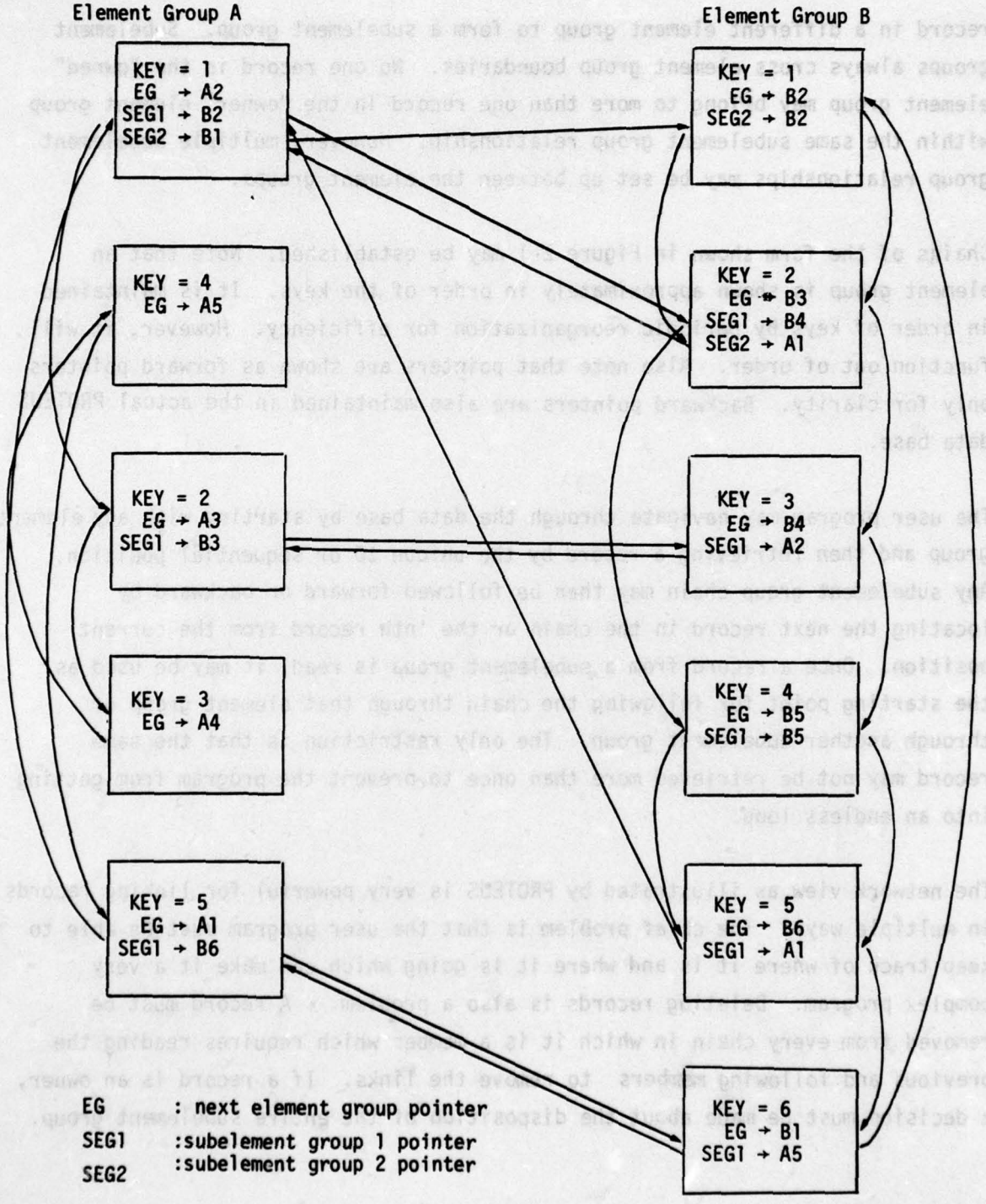
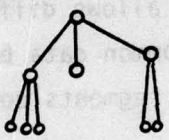



Figure 2-1. Sample PROTEUS Data Base

In general, the network view of the data base has very powerful retrieval characteristics. However, unless it could be implemented in a logical way without using the physical linked list organization, it is not good for a data base with lots of update activity or with unknown queries. Building a subelement group in the PROTEUS system means building chains in the physical file. The chain requires space in the file and time to link and later unlink. Unless it is a chain that will be used often, building it is not justified. If there is no subelement group defined to answer a query, the answer will be more expensive to find than if the file had a serial organization.

### 2.3.2 Hierarchy

A hierarchical view of a data base is diagrammed as an upsidedown tree.



The top is always a single logical record or segment. This segment can be the owner of one or more segment types which can in turn own one or more segment types.

IBM's IMS is a classic example of the hierarchical data base view. In IMS terminology, a segment is "a data element of fixed length, containing one or more logically related data fields [17]. A logical data base record is "a set of hierarchically related, fixed-length segments of one or more segment types...The logical data base record is always a hierarchical tree structure of segments" [17].

A sample IMS logical data structure is the skills data base as shown in Figure 2-2. A sample data base record, based on the structure, is illustrated in Figure 2-3. The root segment contains a single "parent" record with a specific skill. There are three occurrences of the name segment because three persons have the skill of artist. ADAMS and JONES have both experience and education but SMITH has only education. There are four segment types in this example, but the logical record consists of nine segments because of multiple occurrences.

The top segment or owner segment is called the parent. The segments below it are child segments. Segments on the same level are called twin segments.

Multiple logical data bases can exist which contain some or all of the segments from another logical data base. This allows different programs and different users to have their own view of the common data base. Figure 2-4 illustrates a second logical data structure using segments common to the structure in Figure 2-2.

The data base is defined by a definition program. No physical structure or access path is compiled into a program. The user program can give commands to retrieve a unique segment by key, get the next sequential segment, or add, delete, or change a segment.

The hierarchical view of the data base is quite a common one because it is easier to program than a network view. It is also satisfactory for most user views of a data base as long as multiple hierarchical views can be imposed on a data base. There are, however, some applications where a network view is more realistic.

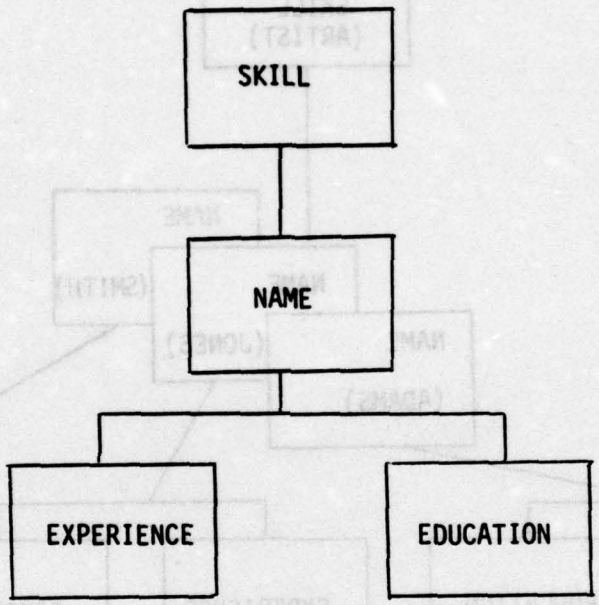



Figure 2-2. Logical Data Structure

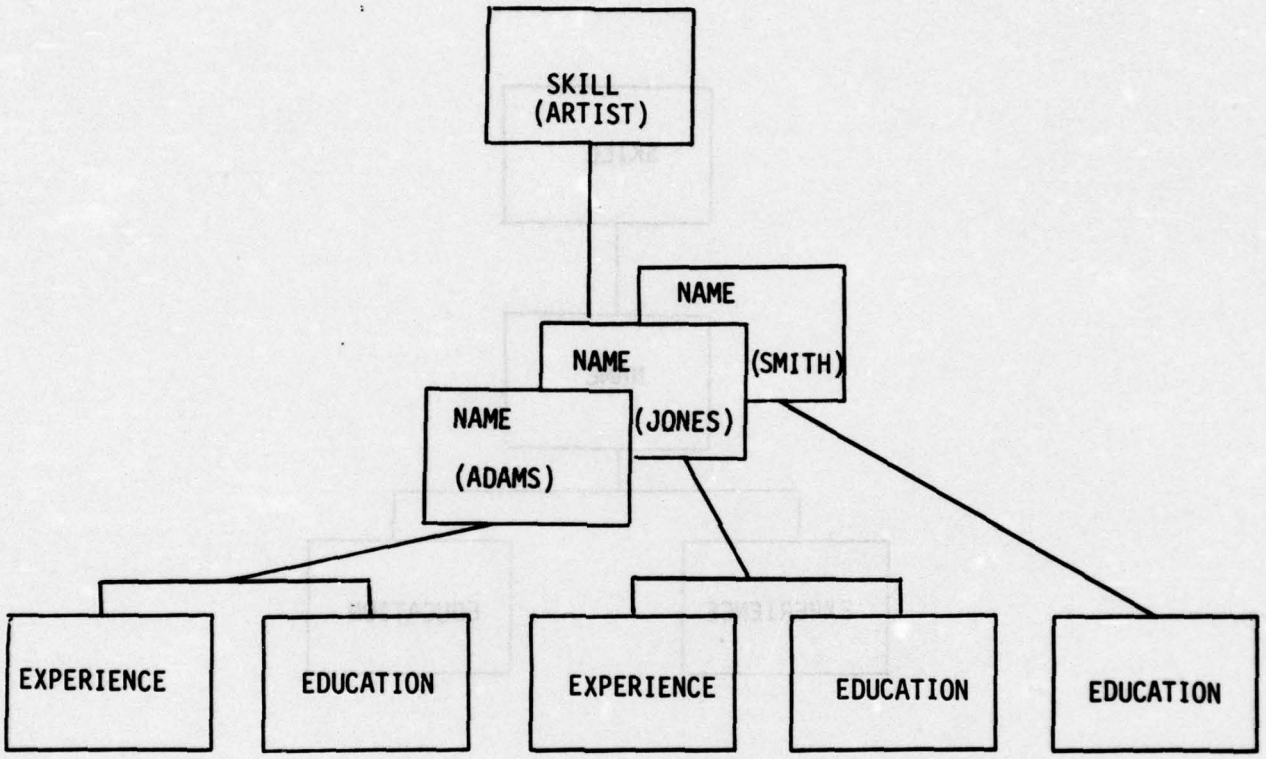



Figure 2-3. Sample Data Base Record

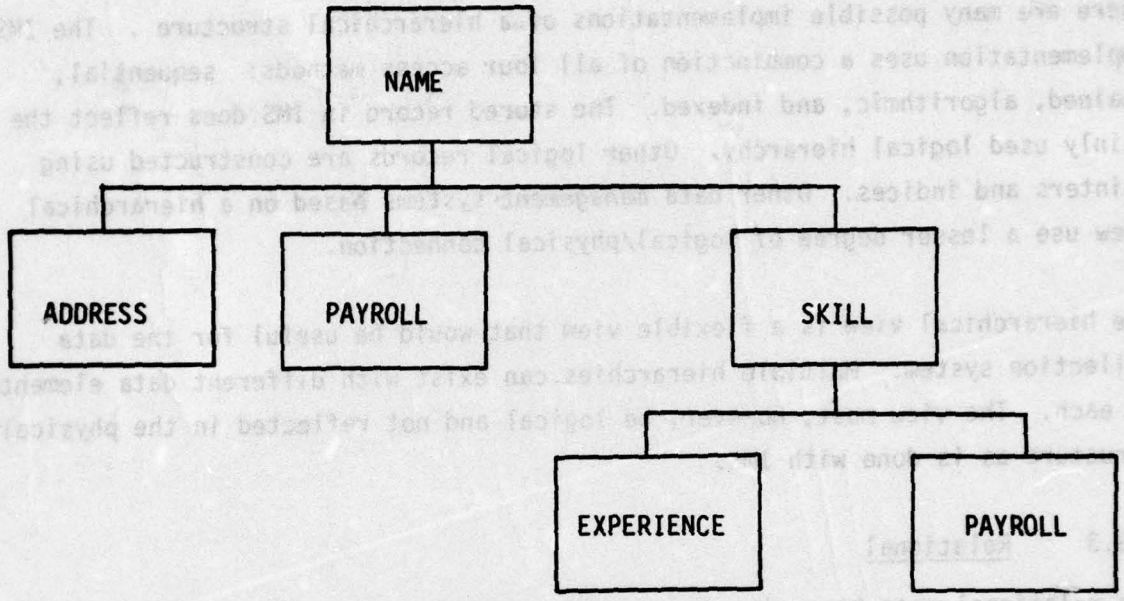



**Figure 2-4. Logical Data Structure**

There are many possible implementations of a hierarchical structure . The IMS implementation uses a combination of all four access methods: sequential, chained, algorithmic, and indexed. The stored record in IMS does reflect the mainly used logical hierarchy. Other logical records are constructed using pointers and indices. Other data management systems based on a hierarchical view use a lesser degree of logical/physical connection.

The hierarchical view is a flexible view that would be useful for the data collection system. Multiple hierarchies can exist with different data elements in each. The view must, however, be logical and not reflected in the physical structure as is done with IMS.

### 2.3.3 Relational

The relational data base view is that of a table or array [6, 14]. Unlike the network and hierarchical views which have evolved from physical data base structures, the relational view was conceived as a logical view and possible implementations are being explored.

A relational data base consists of sets of n-tuples. An n-tuple is analagous to a segment in IMS terms. The n-tuple consists of one or more domains which are analagous to fields. The domain name is the column heading of the logical array and each n-tuple is a row. Every n-tuple has unique values and a unique key consisting of as many domains as necessary to make it unique. A domain never repeats; if it does, the relationship is not "normalized."

Normalization is best understood through an illustration. Assume that an n-tuple exists with four domains:

S#, SNAME, CITY, STATUS

If STATUS is dependent upon CITY rather than S#, that is, a given STATUS is always associated with the same city, a new relation should be made:

S#, SNAME, CITY
CITY, STATUS

Now each non-key domain is functionally dependent upon the key and a value in the relation is not derived from another value in the relation. This concept is stated by the non-relational people by saying that a given piece of data should be stored in only one place.

In order to avoid repeating data, a separate n-tuple exists for each separate occurrence. In the case of the skills data base in the hierarchical examples, a separate n-tuple would exist with NAME, EXPERIENCE for each separate experience for each employee if name is the key. Other data that is not repeating would belong in a separate n-tuple with name as the key.

So far, much work in the field of relational data bases has been expended in the area of languages to access them. Codd proposes a relational calculus and a relational algebra [14]. Both are very mathematically oriented and not directly useful for the average programmer or user. There are, however, some more user oriented languages being developed which are based on the theory of relational algebra. SEQUEL is a language that is being developed by IBM; it uses an English language type syntax [1, 4]. CUPID is being developed at UC Berkeley; it uses a query diagram on a CRT device [22].

Work on implementation is going on at the present time, mainly at universities and other research centers. The internal physical organizations used are the same as those used to implement a hierarchical data base. The key is either indexed or used for algorithmic access to a direct file or index. Codd believes that the simple logical representation of a relational view should mean a simple physical data structure, but that has not yet been proven. Compared to a hierarchical data base, records in a relational data base can

all be fixed length which adds some simplification, but more record types are probably required with correspondingly more indices to keep track of them. Also, for some applications, the relational view may not be simpler even though the data structure is simpler because n-tuples are independent without showing the user a structure. There are applications when the "natural" view includes repeating data or a hierarchy of data.

The relational view is a valid third view of a data base. It may be a good view for the data collection project because it allows many views without imposing a data structure on the view. The problem lies in whether or not it can be efficiently implemented.

## 2.4 EXAMPLE OF A DATA MANAGEMENT SYSTEM

The concepts that have been discussed will now be drawn together by describing one data management system and its implementation. Simscript/PDQ, a system developed and used by PRC for internal projects, was chosen because it uses a variety of techniques and employs the type of data base recommended by the CODASYL committee [5].

The basic unit in PDQ is an entity [23]. An entity is a real world thing - a person, a place, a happening, or an organization. All entities of the same type have the same attributes which can contain single values or arrays of values. The template for entities of the same type is called the entity-type. Entitytypes are defined via a data definition language which stores the definition in a library for inclusion in application programs at compile time.

An entity may own sets containing entities of other entitytypes or it may be a member of a set owned by some other entitytype. Every entity of a given entitytype belongs to a system set which contains all entities of one type. All set relations must be defined in advance in order to allow space in the records for pointers if necessary.

There are two basic kinds of sets. The ranked set is maintained in order of the keys and requires no space in member entities which allows a member entity to belong to more than one owner. The pointer sets are maintained in order of membership through pointers which means that an entity may only belong to one owner. The pointer sets may be organized as LIFO (last in first out), FIFO (first in first out), or LF (last or first).

The user views the data base as a network. Each entity is retrieved either sequentially or directly from the set of all entities of the entitytype. Other entities owned by it may then be retrieved.

The logical view is much the same as the PROTEUS logical view except that members may be owned by more than one owner. The implementation of the two systems is very different. The PROTEUS system uses chained access to a single linked list file. The PDQ system uses more varied techniques.

Each entitytype resides in a separate file of fixed length records. Each file is a serial physical file with a linked list of unused blocks running through it. A new record is either written in an unused slot if one is available or else the record is added to the end of the file.

The new record is then filed in at least one system owned set. Every entity of one entitytype belongs to one set in prime key order. Secondary keys may also be used to file all or some of the entities in another order. For example, all employees in an employee file could be filed by employee number as well as by name.

The new entity is also filed in any other sets that are not "system owned" but are owned by an entity of another entitytype. If the set is a ranked set, an index is used just as it is for a system owned set. The owner record contains a pointer to an index record or chain of records which contain the keys on which the set is ranked and the relative disk address corresponding to each. If the set is a pointer set, the new record is linked to a chain by pointers to the next and/or prior record (depending upon the type of set). The head of the chain resides in the owner record.

Index blocks for all system and ranked sets reside in a separate file that has linked list organization. All index blocks for one set are chained together in key order. The head of the chain is found in an owner entity record or special system owner record.

This example illustrates the combination of physical organizations and access methods that can be used to implement a data management system. It also illustrates how two systems, PROTEUS and PDQ, can use a similar network view of the data base and yet be implemented very differently.

A data management system is more than just an access method. It involves data base access routines which, in the case of PDQ, are subroutines called by an application program. It also involves other programs to perform system functions. PDQ provides a program to build a directory, a PL/I preprocessor to convert English-like commands to PL/I subroutine calls, a program load generator, a "SUPERDUMP" program to dump and restore a data base, and a chain verification program.

## 3. DATA MANAGEMENT FUNCTIONS

A true data management system is much more than an organization of a data base or an access method. It also includes some or all of the following functions: data definition, backup and recovery, restructure capability, security routines, statistical output, a query language, and an access scheduler. Some of these functions may be provided either by the operating system or by the data management system. For purposes of this discussion, it will be assumed that they are all data management functions because of the variation of the functions provided by operating systems.

The functions of data management systems are categorized for purposes of discussion into common, batch and interactive functions. In some cases the lines aren't completely clear but these categories provide a general organization.

The amount of variability in a function will be discussed. Some functions can vary from completely general purpose to very application specific. The general purpose variation is harder to implement, less efficient, but more flexible. Special purpose functions are more efficient, easier to use, and easier to implement but may be harder to adjust to changes in requirements. The trick in selecting the implementation of each function for a particular data management system is to get as specific as possible without losing necessary flexibility.

### 3.1 COMMON FUNCTIONS

The following functions are common to data management systems whether interactive or batch. Not all systems include all functions.

### 3.1.1 Data Definition Language

The minimal data definition language is used to specify to the system where data items reside in the data base as well as the coding type and length of the data items. Many other features can be added to the data definition language such as editing rules for input (array bounds, valid values, required fields), an encoding algorithm or table, a logical data base structure, access authorization, and aliases for data item names. How and by whom the data is used could also be included.

The data definition programs read the user inputs that define the data base and build a directory which describes the data base. There are two basic points in time when the directory may be read. The data structure may be compiled into the programs that access the data base. Either a preprocessor or a compiler accesses the directory. The data structure is then said to be "bound" at compile time. The data base cannot be restructured without a recompilation of all accessing programs.

The directory can also be accessed at execution time. This is satisfactory for an interactive system which is interpreting user requests on a line by line basis. The advantages are not quite so clear cut for a batch system. A program that must be fast and is frequently run is much more efficient if the data structure is bound at compile time so occasional recompiles will be paid for by the better execution times. On the other hand, infrequently run programs or programs that run on historical data perhaps should be bound at execution time. In the case of the data collection system, if there are batch programs written to process historical data it would be advantageous to bind at execution time so that records could be restructured over the years as the collected data changes without converting historical records to the new format. On the other hand, current project management programs that are run regularly might be bound at compile time.

### 3.1.2 Data Base Create, Update, Delete

System routines must be provided to handle changes to the data base in the form of creates, updates, and deletes of entire logical entries or portions of entries. The routine accepts user inputs, verifies the input, and makes the change to the data base.

The data base inputs can be passed to the change routine in one of several ways. The call may be compiled into an application program or the call may be made in a user query language that is interpreted on-line or in batch mode. The data base input may be formatted as a record or it may be self-defining with keywords or other descriptions. Input fields might also be provided interactively as requested by the data management system.

Editing of input may be done by the user application program or it may be done by the data management system based on information in the directory. Simscript/PDQ uses a combination of these two techniques. An offshoot of the directory definition is a source program to edit input. This program is then compiled and stored in a library. The advantage is that editing is much faster with a special purpose program and, in this case, the user needn't code the program. Special purpose programs are only effective though when inputs are in a standard format. Highly variable input requires a flexible, general purpose program that consults a directory.

The change to the data base may be made at the time that the input is accepted, or it may be batched for later processing. Immediate updates mean that retrieval programs have access to the most current data. The chief problem, however, is backup. When continual changes are made to a data base complicated procedures must be followed in order to recover from a program or hardware failure. Another problem is keeping track of changes when multiple users are active. A single change as seen by a user may mean multiple writes of records in order to adjust chains and indices or write multi-part physical records.

If more than one change affects the same set of records, inconsistencies can be introduced. Both the problem of backup and of multiple updates will be covered in a later section. There are solutions to the problems but the solution adds system overhead that might not be necessary if changes can wait until night or at least until system lulls.

Robert Rappaport has designed a system to batch changes but they appear to the user as though they are made immediately [28]. Changes are accumulated in a file as they are made and applied at night or during a slack period. Every access checks the change file first and then the true data base. Because the change file is indexed, only one or few accesses are added to each record retrieval. There still is a cost associated with this method and whether it is justified depends upon the importance of immediate updates. They may not be needed in the data collection system.

### 3.1.3 Data Base Retrieval

The retrieval function is a major function of a data management system. The retrieval function can vary from calls for a record to calls for a report where the user does not know anything about the fields or records. In an interactive query system, the user generally requests the fields by name and has the capability to enter other commands that format or qualify the request.

It is likely that a very flexible query language and report generator will be needed for the data collection system because the requirements are not known in advance. Some special purpose retrieval programs may also be needed to handle regular reports in a more efficient way than can be done with general purpose programs. These special purpose retrieval programs need to be able to request data at a logical entry level by entry key or values of one or more fields in the entry.

### 3.1.4 Data Base Integrity

Every computer system experiences problems with hardware failures and occasional program halts. We are only discussing the errors that cause obvious, immediate halts such as I/O errors or storage overflows; program errors that cause erroneous results are another problem that will not be considered here.

The magnitude of the recovery problem depends upon whether or not updates are being made. If no updates are being made, the system can locate a good starting point for each user and restart the operation. In an interactive system the user could be requested to key in his request again. In a batch system the request must be saved but data base retrieval could be repeated if intermediate results or the place in the retrieval sequence is lost.

Restart becomes more complicated when updates are being made. Chains may be broken or indices left pointing to nonexistent records. Several different methods can be used to recover depending upon the importance of accuracy, the time that the system can be down, and the amount of money that is spent to buy integrity.

Accuracy may not be an absolute requirement. If a data base is a large data base of samples for statistical purposes, the loss of a record is not a catastrophy. The system could make an attempt to retry the update with the knowledge that it might leave something hanging or lose a record. Periodically a clean up program would have to be run to drop dangling pointers.

A more common solution is to take periodic data base dumps and save updates between dumps. When an error occurs, the data base is restored to the state when the dump was taken and then all updates are reapplied. While this is being done, the system is generally not available to users. The cost of dumps must be weighed against the cost to the user of waiting. A large data base may

take a long time to dump which requires computer time and generally users being locked out. The farther apart the dumps, the longer it takes to recover because more updates must be applied.

Selective dumping of only files where changes are being made can make restart faster. Task/Master [33], a data communications monitor and control system, uses selective logs by file and terminal user. Some files may be logged and restored while others are not. Users who are doing purely retrieval do not add to the overhead of the checkpoint facility but users who are changing the data base have all activity saved. Application programs can set checkpoints at critical points in processing rather than have the whole system checkpointed at some interval which is not related to system activity. Messages are also logged so that reports in transmission can be retransmitted. Critical system tables are kept on disk to protect them in case the CPU tables are lost so that knowledge about the status of the various users is not lost.

A different and costly way to maintain the data base integrity is to keep duplicate data bases at different sites. If the hardware fails at one site, the system can be shifted to the backup copy. This allows minimal down time but it costs a lot. It is not going to be cost effective for a data collection sytem.

Until it is decided whether on line updates are necessary for the data collection system, the type of backup and restart cannot be chosen.

Even with the best of systems for backup and recovery, some programs or controls should exist to occasionally verify that the data base is accurate. These checks are often considered debugging tools, but they have a place in an operational system too. Counts of creates minus deletes could be compared to entries in the data base. Index blocks could be read to verify that all records are present and that the record is filed correctly. Chains could be followed to verify that records on the chain should be there and that the

chain doesn't end prematurely. The system must be designed so that these integrity checks are possible. Each record must carry the information necessary to determine its set memberships.

### 3.1.5 Security

Computer systems must be protected from threats from authorized users who accidentally threaten the system and from unauthorized uses who actively attempt to break the system [27]. The threats take the form of unauthorized retrieval, unauthorized modification, and denial of service to users. The threats can be directed at the computer hardware, the operating system, or the access method.

Threats directed at the computer hardware are not covered here. It is assumed that the computer center is physically protected and that unauthorized personnel can not walk away with disks or tapes containing the data base. No matter how secure the operating system or data management system, a thief can find out about the contents of a disk or tape that he takes with him to his own computer. There are stand alone dumps that can be run if the density of the tape is known or if a proper disk drive is found.

The operating system is the second loophole. A secure operating system must have the ability to obtain the identity of all users so that a user cannot falsify his identity, it must be able to prevent data base access except through a data access monitor, and it must be able to protect the users' data from all other users [30]. According to a TRW study of operating systems, such a thing does not yet exist [27]. The CDC KRONOS operating system modified to add security features and the Honeywell GCOS III secure Version for WWMCCS measure up much better than the standard IBM OS/MVT, but they are not completely secure.

A data access monitor has been proposed as a module that can be included in a data management system to control access to the system [30]. The monitor has a file of users, their security classification, and their need to know. It prevents access to data that has a higher classification than the user and assigns classifications to written data based on the highest classification read by the user.

The data collection system will not have data that contains a military classification but data will contain a user access classification. A user should b able to access other user's data only in a statistical form that will prevent him from knowing details about a single other user. A user's data will need to be change protected from other users and from unauthorized personnel within the user's own group. There is a danger that a programmer may want to make his own productivity look better or that of some other programmer look worse.

A data management system can provide some security features over and above those provided by the access monitor. ADABAS is quite secure from stealing data by dumps that bypass the access system because the data is held in fully inverted files so there is no single record for one entry to look at in a disk dump. Data is compressed in the index and also scrambled which makes it difficult to view. Each user has a cipher key that must be supplied to unscramble the data; the system is therefore as secure as the cipher key [7]. A problem with a single cipher key is that the system doesn't work well for multiple users that are allowed to see overlapping subsets of the data base.

A fairly unique requirement of the data collection system is that user perceived security is as important as actual security. If a user does not believe that the data base is secure, it will be difficult to get valid inputs from him. Whatever security features are chosen, they must be visible.

### 3.1.6 System Accounting

The system must keep track of resource usage by user if the cost of the system is to be allocated to the users based on usage. This function may be performed by the data management system, the operating system, or both. Some possible variables are the amount of core used, the system connect time, the CPU time used, the number of devices mounted, the number of I/O accesses, and charges for specific programs used.

In the case of the data collection system, it might be advantageous to allow users to run free while adding to the data base but charge for research or project management usage. Some amount of project management work might be granted in return for data entered but usage above this amount could be charged. The charging algorithm should be designed to encourage users to submit data to the system but discourage unnecessary accesses that would overload the system. Even if RADC assumes part of the cost of the system, some usage charge would prevent excessive system usage.

### 3.1.7 Restructure

There are two reasons for restructuring a data base. The first is to reorganize the storage location of existing records and the second is to change the logical and physical data base structure.

Some access methods require periodic restructure to reorganize the files for more efficient access after additions and deletions have been made. For example, the IBM ISAM access method uses overflow areas for added records. As the overflow chains get longer, access times increase. Occasionally the file must be copied to a sequential file and then reloaded. PROTEUS, the SDC network access method, also needs reorganization to keep records from the same element group together in key order in order to speed up sequential access. Nothing can be said about the needs of the data collection system independent of the access method that may be chosen.

Restructure may also be done to change logical and physical data base structure where the logical change requires a physical change. The more dependent the physical structure on the logical structure, the more likely a restructure will be necessary. In a complete data management system, it should never be necessary to restructure in order to add or delete data fields from one user's view or subschema. It may be necessary to restructure in order to move a data field from one logical record to another or to change the maximum number of occurrences in a repeating group.

The restructure capability for the data collection system will depend upon the type of data structure and the amount of logical/physical independence. The system will not be static and there is a certainty that there will be an evolution in the data that is collected. The changes may be handled either by a binding at execution time or by easy restructure capabilities or both. Structure binding at execution time is more costly in execution time but saves restructure and possible recompile time. These costs will have to be weighed.

### 3.1.8 Core Management

The data management system and operating system or both need to provide a core management system for buffers and other variable storage usage. One possible division of labor is for the data management system to request core based on the number of users or some system parameter and then do its own buffer management within the allocated core.

Fragmentation is the bane of core managers; it occurs when areas of various sizes are allocated and freed until lots of little pieces of free space are available but no large areas remain. This is sometimes avoided by requiring fixed length buffers but that may be a severe restriction. TASK/MASTER has a good approach; it uses a dynamic allocation system which can compress the storage area by moving the buffers and adjusting the addresses of the buffers [33]. This must be done with care to avoid moving buffers while I/O is occurring.

Virtual storage management can be done through hardware, the operating system, or the data management system. A fixed amount of real core is allocated for variable storage. When it is all used, pages are written to working storage based on least used or longest time unused or some other algorithm until they are accessed again. The user program is never aware of the fact that the area is not in real core.

### 3.1.9 Reporting

Reporting covers several reports about the system that are of interest to the people who administer the data base as opposed to the users of the data base. The reports are actually supporting other functions but they will be discussed here as a separate function.

Security reporting provides information about possible security violations. The report may include logon attempts that fail, file accesses that fail, program requests to read working storage before writing it, and full memory dump requests [30].

Data base status report needs depend upon the data structure chosen. In any system, the loading must be known so that more storage space can be added or existing space can be cleaned out in advance of an overflow of storage. Other useful information might be the length of chains, the number of records out of the home bucket, the deviation between physical and logical order, and the access activity for a piece of data. This later report is useful when selecting records for an inactive file or when reorganizing a data base to put more active data in faster access locations such as at the head of a chain or in core rather than on disk or in a index.

A data base monitor is a software or hardware system used to produce activity reports for a data base management system. It can report both on resources used and wait times for various resources in order to locate bottlenecks and improve access times by pointing out areas of the programs that can be improved or inefficiencies in the data base structure.

Some or all of these reports may be desirable for the data collection system.

### 3.1.10 Utilities

Utilities are access method dependent. These are used by the data base administrator rather than the user to check data base integrity, to restructure the data base, to optimize it, to unload a backup copy, or to purge unneeded information.

## 3.2 BATCH SYSTEM FUNCTIONS

A batch system needs no special functions over and above the common functions. One function that may be desirable is an automatic job scheduler to release batch jobs to the operating system job queue based on a scheduling algorithm such as time, another job complete, or input data present. In other systems, batch jobs are submitted by people: users or system control personnel.

It is possible that single users will use a batch system for project management reporting. This might involve a standard job stream. However, most of the computer runs in the data collection system are likely to be special requests or single jobs run as a result of input being present and therefore a sophisticated scheduler would not be needed.

## 3.3 INTERACTIVE SYSTEM FUNCTIONS

An interactive system requires a monitor program to schedule multiple users and to interact with the user at a terminal. If the data collection system has interactive capabilities these functions must be supported.

### 3.3.1 Scheduling

Interactive systems are generally multiple user systems. The handling of the multiple users may be done by the data management system or it may be done by the operating system. In either case, there are some basic jobs to be done.

Multiple users must be scheduled so that each user is given good service. Generally, multiple queues are maintained for the various wait states such as waiting for I/O, waiting for a user response, or waiting for the release of a shared resource. While one user is waiting another is being serviced. A system of priorities is set up so that all users are serviced equally or priority users are given priority service or priority functions are serviced first.

The chief problem that a scheduling algorithm must solve is the prevention of deadlock; this situation exists when neither user can proceed until the other releases a shared resource. A related problem is the prevention of conflicting updates; user A reads record 1, user B reads record 1, user A writes record 1, then user B writes record 1. Both problems are generally solved by having the accessing program announce its intention for the record at the time it is read. Two read only reads are allowed simultaneously but two reads with intention to write are not. All records to be involved in a multiple record update may be read and held as a unit to prevent deadlock.

### 3.3.2 User Coaching

Interactive users may carry on a dialogue with the system. In a batch system, some errors cause a message followed by a denial of the activity requested. In an interactive system, the user is coached with requests for clarification and instructions. This puts a greater burden on the application program to try to understand a user's request.

## 4. SYSTEM HARDWARE

The system hardware discussion covers three basic aspects of hardware: the storage media for the data base, the devices used for data entry, and a word about the processors. A general survey will be made of the types of hardware that might be applicable to the data collection system. Each device is evaluated on the criteria of flexibility, security, cost, and growth potential.

### 4.1 DATA BASE STORAGE MEDIA

Data may be stored on a variety of devices. The basic types are the serial devices such as tape and the devices which may be accessed randomly such as disk or drum. The distinction blurs with mass storage devices which combine features of tape and disk. On any of these devices, the data is grouped into some physical readable block. The block may be fixed or variable length within the constraints of the device.

The following discussion lists manufacturers and prices. These are used only to give an idea of relative prices and hardware capabilities. This paper is not recommending manufacturers or quoting firm prices. Auerback or Datapro are better, more up to date sources for hardware details [2, 13]. Prices must be obtained from a vendor because there is much fluctuation and quantity prices are often available.

### 4.1.1 Serial devices

Data on tape must be accessed and written serially; a tape is either read or written but may not be updated without copying the entire tape. These two features eliminate tape for applications that require random access to records or random continuous updating unless the access or updates can be batched and processed in the order of the tape periodically.

Retrieval of a record from tape is slower than retrieval from a direct access device (with wide ranges of transfer rate for both classes of devices) once the record is located. For example, the IBM 2420 drive can transfer 160 or 320 KB (thousand bytes per sec) depending upon the model. The tape speed is 100 or 200 inches per second. This compares favorably with the transfer rate of the IBM 2314 at 312 KB. The IBM 3330 beats any tape drive at 806KB [2].

The chief advantage of tape is the cost of storing data on tape, whether on-line or off-line. To get an idea of relative costs, as of 9/75 one vendor charges $26.25 per month to lease an IBM 3330-1 which contains a maximum of 100 megabytes (million bytes). A reel of tape which contains 2400 feet and can be written at 1600 BPI (bytes per inch) costs $15.00 to purchase, and can hold a maximum of 46 megabytes.

A tape can be protected by physically locking it up. Generally one tape has one file or a series of files belonging to one user and the tape does not remain mounted when the file is not in use. This makes it more secure than a direct access file that is on a pack with other files, and may therefore be mounted when the file is not in use. Tape files are subject to more physical problems than disks (dust, stretches) but both must be backed up in order to provide recovery in case of physical damage or program halts. The backup of tape files is simpler than the backup of disk files because new records are not written over old.

Tapes may be used in the data collection system for historical data or raw data that has been summarized. It is efficient for storing large volumes of data that will be collected but which need not be accessed on-line. It is not a good medium for current project management data or for files that are to be browsed.

### 4.1.2 Direct Access Devices

The term "direct access device" refers to the class of storage where any record can be accessed directly if the address is known. This includes a wide variety of disks, drums, and data cells, all of which may be called DASDs. With many manufacturers in the field, the announcements of denser, faster systems at lower costs per byte are being made at a rapid rate.

The size, speed, and price of DASDs vary widely. At the small end of the scale are the cartridge disks and floppy disks that are primarily designed for mini-computers. These small disks might be useable for temporary storage for input data but would not offer enough storage for a large data base.

At the large end of the scale, there are many disk packs that can store 100 or 200 megabytes per spindle with 9 spindles per drive. Packs are removable and the arm moves to locate the addressed cylinder. The IBM 3330-11, CDC 33302, ITEL 7330-11, CALCOMP 1035, Memorex 2670, AMPEX DS 331 and TELEX 6330 are all large capacity, fast devices that can store up to 200 MB. There is price competition with various purchase or lease arrangements. All of these are plug compatible with the IBM 3330 [13].

Honeywell offers three types of removeable disk storage, The DS 181 is the smallest with 27,648,000 six bit characters per pack. The drives hold three packs and up to 16 drives can be connected for a total of 443 million characters on line. The DSS 190B packs each hold 117,903,360 six bit characters for a total on-line capacity of 1.88 billion characters. The

largest system is the DSS 0450 which can hold 235 million characters per pack. Up to 16 or 32 drives can be attached depending upon the channel. This gives a maximum capacity of 3.7 or 7.5 billion characters [13].

The non-removable, fixed head systems offer extremely fast access. Burroughs has several systems. The B9473 series has storage capacity of up to 600 million bytes with a transfer rate of 575KB. The CDC 7638 holds 800M with a transfer rate of 6000KB [2]. IBM has announced the 3350 and 3344 [11]. A 3350 module contains two spindles, each capable of storing up to 317 megabytes of data. The average seek time is 24 nsec compared to 30 nsec on the 3330. The 3344 contains 280 megabytes. Both the 3344 and 3350 contain 1 megabyte of fixed head storage per spindle. The first 3350 drive in a string costs $62,500 and subsequent drives cost $49,500.

An intelligent disk system has also recently been announced by IMS Associates [26]. The IMSAI 108 contains a CPU and five or more microprocessors. The host CPU requests a data field and the 108 controller does the indexing, searching, and deblocking. Single spindle units are priced at $29,500.

Disks are expensive for storing huge volumes of data but it would be cost effective to use a disk for any data that is frequently accessed or updated or that is randomly accessed. Searches through large volumes of data to look for relationships or to answer questions can be done more efficiently when the data can be indexed and accessed by key rather than by searching it sequentially which is necessary if the data base is on tape.

### 4.1.3 Mass Storage Devices

The term "mass storage" is ambiguous because it has been used to refer to small disks attached to minicomputers, core storage, or direct access devices. In this paper it is used to refer to a class of devices that combine the storage capacity of a tape library with the direct access capability of a DASD.

There are five manufacturers in the mass storage field. Each has a slightly different product which will be described and then comparisons will be made.

The AMPEX TBM consists of a store of 2 inch wide magnetic tape strips that can be read and the data transferred (staged) to a DASD when it is called for. The storage control processor controls the access to the tapes; it is a PDP-11/45 computer which manages communication with the CPU's, manages the file directory, and locates requested files. The external channel data processor is another PDP 11/45 which is used to transfer data between the DASD and the tape [3]. Ampex claims that the storage costs are $.50 per month per million bytes compared to $6.50 per month per million bytes on a 3330 [9].

The CALCOMP ATL is an automatic tape library system. The system uses conventional magnetic tape which is housed in a storage unit. A reel selector picks out the requested reel and mounts it on a tape drive. It can then be read as a tape or transferred to a DASD under control of user written software. This function is not provided with the system as in the AMPEX system.

The CDC 38500 uses cartridges for storage. The cartridges are mechanically selected when a file is requested and the catridge is automatically mounted on a read/write station. The user's current DASD system may be used for staging under control of a CDC supplied software package called VDAM.

The IBM 3850 uses magnetic tape cartridges containing 3 inch wide strips which are staged to 3330 disks. The "real" space on the DASD is released on a least used basis when more real space is needed. At the time it is released it is written back to the cartridge only if it was changed. The files on the cartridges can be grouped by the user by activity level, or security level, or by other user defined parameters. The user can keep track of the location of his files and remove those cartridges that contain classified or sensitive information.

Precision instruments UNICON system uses laser recorded tape which is off-loaded to a drum for access. It, like the CALCOMP ATL, is primarily a tape library system but it can be accessed as a DASD with user written software. The laser recording means that the tape cannot be reused. It is therefore expensive for systems involving a high level of update since the tape is equivalent in cost to standard tape but there are applications where the fifty year shelf life is more important than the ability to rewrite a tape.

The five systems can be compared on the criteria of performance, capacity and modularity, media considerations, vendor credibility, and cost. An additional factor would be reliability if it could be measured but there are not enough operational systems to measure reliability. All of these comparisons were taken from reference 3.

Performance Measures

| VENDOR EQUIPMENT | AVERAGE ACCESS TIME (SECONDS) | TRANSFER RATES | | MAX RATE |
|---|---|---|---|---|
| | | NUMBER OF SIMULTANEOUS TRANSFERS | TRANSFER RATE (BY BYTES/SEC.) | |
| AMPEX TBM | 17.0 | 6 | 700 | 4,200 |
| CALCOMP ATL | 15.0 | 2 | 780 | 1,560 |
| CDC MSF | 6.8 | 1 | 806 | 806 |
| IBM | 16.0 | 1 | 806 | 806 |
| PI UNICON | .220 | 1 | 400 | 400 |

The Ampex transfer rate of 4200 is theoretical rather than practical since three of the six transfers must be reads and three must be writes.

The CALCOMP access time includes the time required to locate and position the tape. This varies but an average is given. The transfer rate also varies with the type of tape drive. The rate given is for a tape speed of 125 inches per second on 6250 BPI tape.

Capacity and Modularity Measures

| VENDOR EQUIPMENT | STORAGE CAPACITY IN BYTES X $10^8$ | | |
|---|---|---|---|
| | MINIMUM | INCREMENT | MAXIMUM |
| AMPEX TBM | 110 | 110 | 4,000 |
| CALCOMP ATL | 490 | 490 | 9,800 |
| CDC MSF | 160 | 160 | 1,600 |
| IBM MSS | 350 | 668 | 4,720 |
| PI UNICON | 160 | 160 | 1,280 |

Media Considerations

| VENDOR | Supplier Other Than Vendor | Reusable | Shelf Life (Years) | Trans-ferable | Capacity in HMB* | $ Cost per unit |
|---|---|---|---|---|---|---|
| AMPEX | 3 | Y | >15 | ? | 55.00 | 150.00 |
| CALCOMP | 10 | Y | 10 | Y | 1.40 | 9.00 |
| CDC | 0 | Y | 10 | ? | .80 | 150.00 |
| IBM | 0 | Y | 10 | ? | .5 | 20.00 |
| PI | 0 | N | 50 | ? | 20.00 | 220.00 |

*hundred million bytes

Note that the PI media is not reuseable and is also high cost. The extra number of tapes needed to compensate for nonreuseability would have to be figured into the cost of a system.

Vendor Odering Data

| VENDOR | DELIVERY DATE IF ORDERED 9/1/75 | UNITS ON FIRM ORDER | PRODUCTION RATE PER YEAR | SERIAL NUMBER IF ORDERED 2/15/76 |
|---|---|---|---|---|
| AMPEX | 3-6/76 | 4 | 10 | 8 |
| CALCOMP | 12/1/75 | 5 | >10 | 25 |
| CDC | 9-12/76 | >20 | >10 | 25 |
| IBM | 9/76 | >150 | >150 | >100 |
| PI | 3/76 | 1 | 5 | 3 |

The numbers of orders and the number produced by 2/15/76 are estimates based on published information and conversations; vendors do not provide this type of information.

The low number of PI units produced plus the fact that PI is in Chapter XI Bankruptcy make them a high risk vendor.

Costing Data

| VENDOR | MEDIA COST FOR 500 X HMB | ROUGH COST OF TYPICAL MSS (MILLIONS) | SIZE IN HMB | COST PER HMB * |
|---|---|---|---|---|
| AMPEX | $1,363 | $1.2 | 500 | $2,400 |
| CALCOMP | 3,214 | .724 | 1120 | 646 |
| CDC | 93,750 | 1.6 | 657 | 2,286 |
| IBM | 20,000 | .637 | 350 | 1,820 |
| PI | 5,500 | .560 | 320 | 1,750 |

*HMB is hundred million bytes or $10^8$.

The cost is purchase cost but it is not a completely fair measure since different amounts of software are included. The CAMCOMP ATL appears to be low cost but provides a low level of software support.

The mass storage systems hold lots of promise for the data collection system with the expected large volume of data. Storage costs are lower than they would be on a DASD and yet the data can be made available for direct access when access is requested. The chief problem with mass storage is that it is still largely untested and there is little experience with any of the devices. Hopefully, by the time that the data collection system is implemented, more will be known about the reliability aspects of mass storage.

Mass storage implies a central data base which is less secure than distributed data bases. Each system does allow physical removal of a tape or cartridge so that backup copies of the files could be maintained at a different location.

## 4.2 DATA ENTRY METHODS

Data entry methods will be influenced by other decisions made about the data collection system such as whether the data base will be central or distributed, how fast the data must enter the data base, and how much data is to be collected. Possible data entry methods will in turn influence the volume of data that can cost effectively be collected and the type of form that will be used. This discussion will cover the possible choices of data entry methods and the classes of hardware that are available for data entry.

### 4.2.1 Options

Several options are possible when choosing a method for data entry. The options will have to be weighed to see if they satisfy the other requirements of the data collection system including cost constraints.

The first option is whether data is to be collected manually or automatically. It is likely that both methods will be used. Automatically collected data need not be entered with human intervention which makes it simpler, faster, and cheaper to enter. The only decision involved here is how to transmit it to the central data base.

Manually collected data must be entered either at the source by a programmer or system librarian or it can be recorded on a document for later transcription by a typist or keypunch operator. If the data is entered at the source it can be recorded in machine readable form by entering the data on a terminal or typing it in a form readable by an optical character reader.

If the data is collected on documents that must be transcribed, they can be keypunched to cards, tape, or disk or typed for reading by an optical character reader. This transcription can be done locally or the documents may be sent to a central site.

Editing of input may be done at the time it is entered so that the operator can make immediate corrections or it may be edited at the time that the data base is updated.

The manually collected or automatically collected data may be stored in a single central data base or it may be stored in a local data base. The local data base may be accessed from the central computer or summaries or samples of the local data base may be transmitted to the central data base. If the data base is a single central data base, the input data may be transmitted in raw or machine readable form to the central data base by communication lines or by mail. If the data base is distributed, data entry must be distributed.

### 4.2.2 Data Entry Hardware

The hardware will be evaluated against the background of the options. The cost and availability of various devices will influence the choices that are to be made.

It is likely that a mixture of data entry methods will be used depending upon the equipment available at a remote site. If data is stored in remote data bases and transmitted from the remote data base to the central data base, no standardized methods are necessary. If data is transcribed to machine readable

form locally and transmitted by tape to the central location, there can be a variety of methods as long as the output is a standard tape. Only if data is input directly to the central location does there need to be a small number of methods that are predefined.

#### 4.2.2.1 Keypunch

Keypunches that punch cards have been around for a long time and are widely accepted even though there are now devices that allow greater throughput. Keypunches have been improved in the last several years with the introduction of the buffered keypunch. The UNIVAC 1700 series allows the operator to punch, verify, and correct the card before it is punched. This saves both time and card stock. The IBM 129 is also a buffered keypunch which can store up to six control programs.

Keypunches can be purchased for $3,000 to $7,600 and rented for $69 - $200 per month depending upon the model and options. The buffered UNIVAC 1710 with punch, verify, and interpret is the top of the line at $7,560 [19].

Keypunches would not be a cost effective way to transcribe data at the central site. They might be used at remote sites if there are currently keypunches installed and the volume of data does not warrant new equipment for the data collection system. The card output could then be used to update a local data base or could be transmitted to the central data base by sending card images over a communications line or by reading them to tape and mailing the tape.

#### 4.2.2.2 Dumb Terminals

The "dumb" terminals that will be discussed here are controlled from a CPU that is involved in other activities. In the next section, intelligent terminals that contain their own controller or that are clustered around a controller are discussed. The distinction blurs between a large scale data

data entry system which has a processor capable of running background jobs and a system that runs terminals as its "background" jobs.

The problem with terminals connected to a CPU is that the communications line must be open throughout the session although much of an operator's time may be spent thinking and keying. If the central computer is a long way off, it is cheaper to store data and transmit it at full speed as is done with the key to tape systems rather than keep the line open all day. However, if the terminals are near the computer this need not be a consideration. Then the chief consideration is whether the slice of central CPU time is cheaper than a dedicated mini or the microprocessor that comes with an intelligent terminal system.

The terminals might also be connected to remote user computers. This shortens the communications line but means that many copies of editing programs would have to be written to interface different types of computers and terminals. Telecommunications systems are difficult systems to implement, even for just a few terminals, and therefore it probably would not be feasible to have multiple systems running at remote computers.

Terminals would normally be used for collecting data directly from programmers or program control personnel through an interactive question/response session or through a form to be filled in on a CRT face. Terminals are not designed for keypunchers to use for transcription from documents if there is a large volume of documents although they can be programmed to do fixed field with duplication which is normally used for repetitive keying.

The most basic type of terminal is the Teletype Model 33 terminal. It has a keyboard and a printer to print input and response. When used on-line a program can control editing and response. When used off-line a paper tape can be punched for later transmission; no editing can then be done while the data is being entered. This type of terminal sells for $700 to $1056 depending upon options, and whether a tape unit is attached.

This, and other hard copy terminals, are cheaper than the display type of terminals and also provide hard copy which may or may not be necessary. The Teletype Model 33 is slow at 110 bits per second but other hard copy terminals such as the Memorex 1242 advertise speeds of up to 9600 bits per sec, the maximum speed of a leased line. The price also goes up to $3300 to $4000 [12].

The CRT type of terminal is faster than a typewriter type of terminal and can be used to provide a form for the user to select options or fill in the blanks. The IBM 3270 is a prime example. It can operate as part of a multi-station system with a control unit or it can operate as a remote terminal communicating with a central computer. The remote terminal, the 3275, sells for $6222 to $12072 depending upon the screen size and speed [24]. The problem with the 3270 is that it introduces a new IBM standard systems network architecture which makes it difficult to interface with non IBM computers.

There are many 3270 competitors and some of them do not use the new IBM network protocol which makes them more general purpose. The prices are also lower for the competitors. For example, the ADDS 980A terminal from Applied Digital Data Systems sells for $3200 [12]. The Leir Seigler ADM-3 sells for $995 with a 12 line scope face [12].

The Honeywell VIP 7700 can operate in cluster mode with a controller or directly connected to a Honeywell 6180. It sells for $4860 to $16755 for a display unit with the controller an additional $3035 [13, 24]. Both the IBM 3275 and the Honeywell VIP 7700 would fall into the classification of an intelligent terminal if a controller is attached.

The Teletype Model 40 is a medium speed terminal with a CRT display unit and, optionally, a printer. The display screen holds 24 lines of 80 characters. The transmission speed is up to 4800 bits per second [12]. It is available for purchase from Teletype for $3086 - $6473 depending upon options and

whether there is a printer. It may also be leased from AT&T, offered as Dataspeed 40 service, for $110-$205 per month. A magnetic tape may be attached to a Dataspeed 40 terminal for $110 a month [13].

Terminals also need a modem to connect the terminal to the communications line if a voice line is used. Some terminals have a built in modem with acoustical coupler while others require a separate modem. The modem costs $150 - $10,000 depending upon the transmission speed, type of line, and line protocol. The higher priced modems also multiplex several channels on a single line.

Terminal input probably will not be effective for the data collection system. If the input goes directly to the central data base, communications costs would be prohibitive. If the terminals are connected to distributed local computers, editing programs would have to be written for each type of computer. The one case where it might pay would be the case where programmers are doing development via terminal and data is being collected for project management purposes. A form could be put on a screen requesting results from the previous run when a programmer logs on for the next run. Terminal costs would not be significant if the terminal is also used for program development and the capture of data directly from the programmer would save the cost of a keypunch staff. This is not a decision that can be made on cost alone because the willingness of the programmer to fill in a form on a terminal vs his willingness to fill out a paper form must also be considered.

Terminals may be used for querying the data base. The user would then need access to the central data base and terminals allow fast interactive access. In this case the decision must be made concerning the importance of fast interactive access versus cheaper batch access which can also be subject to greater control.

#### 4.2.2.3 Intelligent Terminals

The intelligent terminals have a local mini-computer to control one or more stations or a microprocessor into the station. This type of system avoids the problems of line protocol and communications cost that dumb terminals have. Data is entered and stored on a disk or tape which can be either transmitted or carried to a central computer. This is sufficient for data entry use but is not sufficient for queries directed to a central data base.

Intelligent terminals can be used both by continuously keying operators and by occasional operators. The continuously keying operator would be a full time operator who is transcribing programmer produced documents into machine readable form. This type of operator generally keys the data in a fixed format with fixed or variable length fields. He can operate faster without looking at the terminal. The occasional operator would be a programmer or program control person who is entering results of runs a few times a week. This type of operator is best served by a fill in the blanks, question and answer, or multiple choice format [21].

There is an explosion of new devices in the key to disk and tape market. They range from devices with the same capabilities as a keypunch to some that are actually a small computer system.

The key to tape/disk devices would have a use in the data collection system if manually collected data is written on forms that are not machine readable or if data is entered via terminal without recording it first. Forms must be transcribed to machine readable form either at the local sites or at the central site. If the data base is a distributed data base, the transcription would be done locally so several smaller, cheaper systems would be needed. Some standardization would still be desirable so that input editing programs could be standardized but this will not be possible if the volume of data

is low. If the data base is centralized, the data transcription can be done centrally by mailing documents to a central site. A more complex system could then be supported cost effectively.

At the simpler end are the standalone units that punch, verify, and store the output of the single station on a tape or cassette which can then be transmitted or carried or mailed to a computer for processing. An example is the Burroughs TC4000 which can be used to enter data to a floppy disk or cassette for later communication to a CPU or another TC4000. Data is typed and may be visually verified on a screen before transmission or storage. It sells for $4945 to $9495 [9].

The Honeywell Keytape stations are individual units which can transcribe data from the keyboard to magnetic tape. The system operates in punch mode, verify mode, or search mode which will retrieve previously entered records. The operator may have two programs to control the format with alpha and numeric fields, automatic field skipping, duplicating, or left zero fill. Fields can also be totaled or checked with a check digit. Once inputs are entered the operator can combine short tapes or transmit tapes to a Honeywell series 200 computer or another Honeywell Keynet station. Prices range from $2500 to $8550 per station for the basic unit [13].

The Mohawk System 2300 is an intelligent programmable terminal. It is designed for source data entry where a programmed series of requests may be displayed for response from a clerk, or, in the case of the data collection system, a programmer. The programmer would enter results of runs at the terminal rather than filling out a paper form. The input is edited, formatted, and stored on a disk or tape for tranmission at night. The disk may also be used to store data for program lookup. The tape could be carried or mailed to a remote computer rather than transmitting it. The price for a controller with a disk, CRT, and keyboard is $10,400. A tape drive, printer, card reader, more core, and more disk storage can be attached. [13].

The SYCOR 250 system can operate as a single standalone intelligent terminal or as a cluster. The user can program checks or formats using a parameterized language called FIL. Accepted data can be stored on a diskette for transmission at night. The smallest single station system with a 480 character display sells for $4840. Multiple station systems cost $2290 for a 480 character display controller and $3440 for each display station [13].

The multiple station key to disk/tape systems use a mini as a controller for several stations. The editing capability is greater, in some cases, than that possessed by the intelligent terminals, but it is necessary to have a large volume of data to support the system. It could be used if all input is transcribed centrally or if other applications could share the cost.

The Sycor 440 contains a mini which can support four to eight keyboards. The processor can control the keyboards in a foreground partition to do editing and formatting of input. A background partition can be used to run programs on previously entered data such as batch totaling, sorting, file lookup or file transfer. It can also be used as a polling station at the central computer to receive data from remote 440s. The system can support 5 to 10 million characters of disk storage plus a printer, card reader, and magnetic tape drive. A programming language, TAL II is provided for writing foreground and background programs. The system sells for $29,090 for a four station system or $38,660 for an eight station system. Leases are also available [11].

CMC offers a large scale data entry system that can grow to 64 key stations. The CMC 1800 also has foreground and background partitions for editing while keying and running programs on previously keyed data. There is a KOBOL language for writing foreground edits and an RPG II language for writing background programs. The price for a system with ten stations is $86,500 [11].

The keyboard programmable terminals are slightly lower in price than the display terminals. These include the Burroughs TC600 and 3500, IBM 3735, Data Measurement's DMC 400, Qualterm X100, Compro 1030 and 6000, and Texas Instruments 742. Prices range from $4,000 to $15,000 with an average of around $10,000 [29].

#### 4.2.2.4 Optical Character Readers

Optical character readers, also called OCR, are used to read characters from source documents. The characters must have been typed except for a few expensive versions which can accept carefully handwritten numbers. Some accept only a single type style (single-font) and others accept several common styles (multi-font). Some allow certain fields to be read by the OCR while others are entered by an operator. Others run with an operator who can enter unreadable characters instead of having documents rejected. The output of the OCR may be stored on disk or tape or it may be on-line to a computer.

An optical character is feasible where there is high volume and the data can be captured at the source by typists who are typing inputs anyway such as clerks taking orders and typing order forms. The case is not as clearcut if handwritten documents must be typed. It may be a bit cheaper to type the documents than to have them keypunched because typists earn less than keypunch operators [13], but there are several problems. Special forms must be designed and purchased and every field must be typed since typewriters have no automatic dup. No editing can be done either to catch errors that are made by the typists. Mailing documents to a central site for reading by the OCR introduces security problems in protecting the documents.

The OCR systems may be purchased for $16,000 to $1 million depending upon the speed and the number of type styles that can be read. The norm for a system that can read 8 1/2 x 11 pages offline and create a tape, read multi-fonts, and operate at 120 lines per minute is around $50,000 [13].

The data collection system would probably not benefit from an optical character reader if the source document must be retyped. On the other hand, if the data is collected by system control personnel who could type it on forms readable by an OCR, it would be more cost effective than rekeying the data. It may not be more cost effective than entering it via a terminal where it could be edited immediately and transmitted within hours.

4.2.2.5 Communications Consideration

Many of the decisions about local vs central data bases or local vs central editing programs depend upon the cost and availability of communications lines. Is it cheaper to input data on an off-line Teletype which goes on-line to transmit a tape or is it cheaper to edit on-line to avoid retransmission of corrected errors? Is the immediate availability of data worth the cost of transmitting from remote sites to the central site rather than mailing tapes? Communication line costs and communication processors are discussed below.

There are two basic types of communications line: the switched network and the point to point dedicated line. Telephone lines are the most readily available network lines; they may be charged at direct distance dial rates or under a WATS line agreement. The dedicated lines may be leased from one of several companies: AT&T, Western Union, MCI, SPCC, Datran, American Satellite Co., and RCA Global. Figure 2-5 shows a cost comparison of some options [13]. The times are for hours of connect time but the telephone lines transmit at a lower rate: maximum 4800 bits/sec compared to maximum 9600 bits/sec for a leased line. Consideration must also be given to the fact that a line may not be needed on a steady 8 hour basis.

A new option is being developed: the private switched network system. This concept offers the flexibility of the telephone lines with a system designed for data transmission rather than voice transmission. Datran started with private leased line service to establish a network which they are now

| | LEASED AT&T 3002 | | DIRECT DISTANCE DIAL $/HR | | | WATS - 1 SHIFT 100% USE | |
|---|---|---|---|---|---|---|---|
| MILES INTERSTATE | LO/LO (1) $/HR | HI/HI $/HR | 9AM-5PM | 5PM-1AM | 1AM-9AM | FULL DAY 8 HOUR RATE | MEASURED (2) TIME RATE |
| 50 | 1.18 | .96 | 7.80 | 5.70 | 4.95 | 3.30 | 11.06 |
| 100 | 1.90 | 1.21 | 10.20 | 6.00 | 6.00 | 3.30 | 11.06 |
| 1000 | | 5.60 | 21.00 | 12.00 | 12.00 | 8.71 | 19.11 |
| 3000 | | 15.37 | 27.00 | 14.25 | 12.38 | 9.74 | 22.17 |

(1) LO/LO or HI/HI is rate based on density of area

(2) Measured time rate has a flat rate for 10 hours per week and an hourly rate over that.

Figure 2-5. Communications Costs Comparison

offering as a switched network service. They use microwave relay stations throughout the United States which broadcast to and from a computer in Illinois. As of the end of 1975, 18 cities are serviced. By the middle of 1976, the plan is to have 30 cities connected to the network [15].

The Datran network is more reliable and lower cost than the telephone network if the transmission is to and from one of the Datran nodes. Datran guarantees transmission accuracy of 99.95% and will not bill for periods of substandard performance. The rate structure is: [8].

| Transmission Speed (BPS) | Channel Rate per 100 mi per min | Monthly Charge Area I* | Area II* | One time installation |
|---|---|---|---|---|
| 2400 | $0.0015 | $130 | $165 | $150 |
| 4800 | $0.0020 | $140 | $175 | $200 |
| 9600 | $0.0030 | $150 | $185 | $200 |

*Area I within 5 miles of Datran center, area II within 40 miles

There is also a 20% discount for night use and a discount for bulk users [15]. These rates put the cost way below the cost of direct distance dial. A five minute call to transmit data at 2400 BPS for 100 miles costs $.08 using Datran and $1.45 using direct distance dial [15].

The big problem with the Datran network is that it does not cover all cities served by the telephone lines. Telephone lines would still be needed to access Datran nodes. If the data collection system needs to transmit from points far from Datran nodes, direct distance dial may still be the most economical method.

There are some other private networks available. AT&T offers Dataphone Digital service which serves 24 cities as of the end of 1975 [8]. The rate is monthly rather than based on use. The rate structure is:

| Transmission Speed (BPS) | Monthly Channel | Access Charge Area I* | Access Charge Area II* | One Time Installation |
|---|---|---|---|---|
| 2400 | $20 + $.40 per mile | $65 | $ 90 + .60 per mile | $100 |
| 4800 | 40 + .60 | $85 | 110 + .90 | $100 |
| 9600 | 60 + .90 | $110 | 130 + 1.30 | $100 |

*Area I within 5 miles of office, Area II over 5 miles

Telenet is a value added carier which runs a packet switching network using the AT&T lines. It expects to serve 30 cities by the middle of 1976. By accessing the lines through Telenet, the user can pay a usage charge plus a flat rate instead of the flat rate charged by AT&T.

Tymshare runs a timesharing service that has a nationwide network of 100 nodes. A user may use the network without the timesharing service by becoming a Tymnet Customer. The user dials the closest Tymnet number and pays direct distance dial rates to the node and Tymnet rates the remainder of the way to the user's central computer.

Mention should also be made of low speed and high speed lines [13]. The low speed network is run by TWX/TELEX of Western Union. The transmission speed is 110 bits/sec compared to 4800 bits/sec for the telephone system. The cost is $12 to $42 per hour depending upon the distance which makes it more expensive than direct distance dial.

High speed transmission can reach speeds of 24,000 bits/sec to 56,000 bits/sec. As of this report, high speed service is available from AT&T Dataphone Digital Service and from Datran. This type of service is designed for communication between computers. It is only available on a private leased line basis.

If any remote terminals are going to be used in the data collection system, the central computer will have to be equipped with a communications processor or software in the central processor to handle the communications functions. These functions include message switching, front end processing for network control, polling, message assembly, data conversion, editing such as decompression, error control, message buffering, message switching, simple message answering, message recording for backup, and statistics recording. There are two communications processors which can be used with the Honeywell 6180. The System 7000 can handle 128 lines and input from a Teletype or Honeywell CRT. The Datanet 355 can handle 200 lines and input from Teletype, Honeywell or IBM terminals, and an IBM 360/20 [13]. Some of the intelligent terminal systems can also be used as communications processors for similar remote devices. For example, a SYCOR 440 can be used to poll and receive from remote SYCOR 440 stations. The incoming data could then be written to tape for hand carrying to the central computer.

Data transmission is an alternative only for transmission of data within the United States. Three carriers, RCA Global, Western Union, and ITT are working on international systems [8].

## 4.3 PROCESSOR REQUIREMENTS

Little will be said about processor requirements in the abstract. The processor requirements will be dependent upon the amount and distribution of the data that is collected, how it is entered, and how it is accessed. The costs of processor capability must be considered as part of the cost of each design option.

This discussion has assumed that the central core of the data collection system will be a Honeywell 6180 or other large computer. Other processors might also be used such as a user computer to update a local data base, a mini to control entry stations, or a front end communications processor to handle data transmissions.

## 5. SUMMARY

In summary, the basic issues concerning the interface between the data collection and the data base storage system are concerned with the degree of centralization, the data base structure, and the data management capabilities that are chosen for implementation.

### 5.1 CENTRALIZATION

If only basic, long-term research is supported using derived data, a completely centralized data base and facility would be adequate. If immediate project management is to be supported, localized data stores and project monitor programs may be added.

The chief disadvantages of the centralized data base lie in the very large volume of key entry for manually submitted data, the time delay and difficulties inherent in correcting errors, and the small benefit that data suppliers receive in supporting the centralized store. It is concluded that machine-readable input be accepted, that adequate editing capability (preferably supported by an "intelligent terminal") be provided at the point of data acquisition to ensure elimination of the majority of input errors, and that the data collecting effort be integrated with and support project management operations as much as possible.

The chief disadvantage of local (project) data bases lie in the difficulty of achieving standardization for many projects and machines, in the cost of providing standard hardware or software to many projects, and, if the local store is on-line with the central store, in protecting the privacy and integrity of sensitive data. It is believed that sufficient advantage in the efficiency of operations and the quality of data may be had to offset these disadvantages and that adequate protection mechanisms can be developed to ensure privacy.

## 5.2 STRUCTURE

It is concluded that it is most efficient for records to be written serially on storage devices and that multiple keys (one or more) be permitted to provide adequate retrieval power. Indices and current records should be held in direct access storage (such as disc) and historic files be kept in secondary storage (such as tape or mass storage). Information linkage will be served if pointers in one record type are allowed to point to records of another type or to an index containing such pointers. The structure should be modular and flexible, as is the data collection system, to permit easy extension and modification.

## 5.3 DATA MANAGEMENT CAPABILITIES

The capabilities that the data management system should have included:

- A physical structure transparent to users.
- Multiple user's views adopted to the logical requirements of the usage.
- Sufficient independence of the physical and logical views of the system to permit ready modification of one without distortion of the other.
- Security keys to control data base access.
- Access to data outside the logical view of the user should not be permitted.

## 5.4 CONCLUSIONS

If it is assumed that the initial operation of the Software Data Repository will be vested in an existing computer, operating system, and data management system, then the results of the investigation of data base and data management system characteristics represent a major advance over currently available software. While a pilot facility may perform adequately in handling a limited variety of data items from a select group of projects and in supporting a limited number of research projects, full scale operation should be fully buffered against the volume and unreliability of manual input forms. Greatest buffering and greatest benefit may be realized by merging the centralized data repository with a standardized project monitor system and local data bases to reduce data volume, create machine-readible input, and support project management as well as basic software research.

## BIBLIOGRAPHY


[1] Astrahan, M.M. and Lorie, R. A., SEQUEL-XRM, A Relational System ACM Pacific '75 Conference Proceedings.

[2] Auerbach Computer Technology Reports 1975 - Input/Output Reports Volume 20.

[3] Bowman, S., Study Report, Data Base and Mass Storage Configuration Options, System Develompent Corporation TM-(L)-5582/002/00, 1975

[4] Boyce, R., Chamberlin, D., King, W.F. III, and Hammer, M., Specifying Queries as Relational Expressions: The SQUARE Data Sublanguage Communications of the ACM, November 1975.

[5] CODASYL Data Base Task Group (DBTG), April, 1971 Report

[6] Codd, E.F., Recent Investigations in Relational Data Base Systems ACM Pacific '75 Conference Proceedings.

[7] Cohen, L., Data Base Management Systems.

[8] The Data Communications User, November, 1975.

[9] Datamation, May 1975.

[10] Datamation, July 1975.

[11] Datamation, September 1975.

[12] Datamation, November 1975.

[13] Datapro Research Corporation, Datapro Research, 1975.

[14] Date, C.J., An Introduction to Data Base Systems, 1975.

[15] Datran sales literature, 1975.

[16] Deutscher, R., Tremblay, J., Sorenson, P., A Comparative Study of Distribution Dependent and Distribution Independent Hashing Functions, ACM Pacific '75 Conference Proceedings.

BIBLIOGRAPHY (cont'd)

[17] IBM, Information Management System/360 Version 2, General Information Manual, GH 20-0765-2.

[18] Lum, V.Y., Multi-Attribute Retrieval with Combined Indexes, Communications of the ACM, November 1970.

[19] Management Information Corporation, Data Entry Today, 1972.

[20] Martin, J., Computer Data Base Organization, 1974.

[21] Martin, J., Design of Man-Computer Dialogue, 1973.

[22] McDonald, N., Stonebraker, M., CUPID - The Friendly Query Language, ACM Pacific '75 Conference Proceedings.

[23] Planning Research Corporation, Simscript/PDQ Manual, April 1971.

[24] Granholm, J.W., Alphanumeric Display Terminals, Datamation, January 1976.

[25] DNA

[26] Product Spotlight, Datamation, September 1975.

[27] RADC, Computer Security Technology Reference Manual, December 1974.

[28] Rappaport, R., File Structure Design to Facilitate On-Line Instantaneous Updating, ACM Pacific '75 Conference Proceedings.

[29] Salzman, R., The Computer Terminal Industry: A Forecast, Datamation, November 1975.

[30] Schaefer, M., Secure Data Management System Preliminary Mathematical Model, System Development Corporation for RADC, TR-74-352, 1975, (786423).

[31] Senko, M. E., File Design - A Practical Approach, Volume 1, File Design Handbook.

[32] SHARE, Share Data Base Project, June 30, 1974.

[33] Sisco, D., Sales Seminar Discussing TASK/MASTER, package developed by Turnkey Systems, October 1974.

☆U.S. GOVERNMENT PRINTING OFFICE: 1977-714-025/86