Researchers assume, I think quite readily by now, that a vast array of computer-based analytical tools will be available to support their research projects. The analysis of data is, assuredly, the researcher’s primary responsibility. Yet the need for equally powerful facilities for the management of data is also evident — certainly to those of us given that management task. At a minimum, the ability to manipulate data for use with sophisticated statistical techniques must exist. Along with a greater reliance on computers in research has come an increase in the amount of data collected and the complexity of relationships inherent in the data. Because of the computer’s power, limitations on project design are less and less based upon data processing constraints. The problems of data integrity and security also add to the demands placed on data management.

The Cooperative Study of Sickle Cell Disease (CSSCD) with which I have been associated is a fitting example. The CSSCD is a prospective study of the clinical course of sickle cell disease. Twenty-seven clinics are participating. During the one and one-half year period, in addition, approximately 800 controls will be followed for a newborn cohort. Data is collected at the individual clinics, but all processing and analysis of the data is done at a single statistical coordinating center. The information collected includes demographic, historical, laboratory and physical data defined by over 3000 variables. It is estimated that approximately 150,000 forms will be processed during the course of the study.

The CSSCD project chose the Scientific Information Retrieval (SIR) [1] data base management system (DBMS) to meet its data management requirements. Our choice of SIR may seem contradictory in regard to my remarks concerning the researcher’s analytical needs when I tell you that SIR has only a minimal set of descriptive statistical tools. However, SIR is almost unique among DBMS in providing a direct interface to statistical software packages. I have taken specific examples from the CSSCD project to emphasize problems which were of major importance in our selection of SIR.

I. DATA STRUCTURING AND DATA RETRIEVAL

I have combined these two topics into one section because the type of data structures which a DBMS supports has a very direct effect on the methods and language of data retrieval. Our concerns here are with both the physical and logical structures provided by the DBMS. We will recognize that the physical organization of data, the actual storage of information, would efficiently map the study data. On top of this physical model is our conceptual view of the study. Regardless of how information is stored we want to view the project data as consisting of a variety of distinct “records” describing different events and collected for each patient. A DBMS should support this logical view and be capable of modeling the relationships inherent in the data structures. The language and mechanisms of data retrieval should also allow us to work with our data at this logical level. The following discussion of the CSSCD project gives examples of these data structuring and retrieval problems.

A. Problem Areas

1. Variation in type and amount of data per patient

The data collected for a patient in our study is defined by age (or protocol) and the course of the disease during the study. While all protocols are managed in a similar fashion: demographic, historical, and physical baseline data are collected; interim visits are regularly scheduled for examinations and laboratory tests; and exit data is collected - data items differ among the protocols as do the frequency of scheduled visits. The variation among patients for acute events and chronic complications is even greater. Patients differ in the number and type of events which occur and the frequency with which any specific event occurs. This study may be extreme in the variation of data groupings among its patients, but the basic outline is certainly common to clinical studies.

It is clear that the data cannot be stored as a single “rectangular” file (i.e., a file where all records are defined by one common format), nor would
we want to conceptually view it that way. This is true for a variety of reasons; for example, storage requirements would be prohibitive, and the impact of even a minor change to the data structure would be severe. This variation in type and amount requires that the data be organized into multiple record types which allow for a varying number of record occurrences. [2]

2. Relationships among data for a single patient

Although we may choose to divide a patient's data into separate record types, the "patient" still remains the basic unit of analysis. This may be obvious, but it has important practical ramifications. If I want to analyze the data for a patient, I do not want to manipulate all these separate record types. A requirement of the DBMS should be the implicit integration of these units.

Relationships between record types which must be supported are of several kinds. Most often records are logically related simply because they represent events occurring at the same time, such as all baseline data, or the second cycle of scheduled laboratory tests and physical examinations. Records are also related in a nested or hierarchical fashion. A typical example of this is the need to access all follow-up examinations which were initiated because of the occurrence of a particular acute event.

3. Relationships between patients

Even though data is usually viewed as belonging to a single patient, there are situations where links must be provided between patients.

4. Matched controls processing

Matched controls are being followed for a newborn subset of the study population, with zero, one or two controls possible for a patient. Newborns and controls are normally analyzed as distinct groups, either separately or in comparison with each other. These types of analysis add no special structural requirements to a data management system. Other analyses, however, require a direct comparison between an individual patient and the associated controls. Such analyses do require more complicated structural pathways in the data base. This is particularly true because the logic of the retrievals, in terms of direction, may be either from patient to control or from control to patient. We would also hope that whatever implementation is available would reduce the complexity of application programming.

b. Cases sharing a common set of data

In the CSSCD study it is common to find two or more patients from the same family. Demographic information for the family is collected only from the patient entering the study first, but must be associated to all patients in the family. Our preference is to maintain a single copy of such data to reduce storage requirements and to avoid the problems involved with updating multiple copies of the same data.

4. Data independence and data retrieval

One of the purposes of data base management systems is to provide data independence. This concept has many aspects, but generally implies that the underlying physical organization of data should be transparent to user applications. A simple example would be the addition of a variable to a record type. Such a change should be invisible to existing applications, and thus have no effect on them. Another aspect of data independence is related to data retrieval. This process is easier and far more reliable when the user can approach the problem as it is defined by the logic of the study, remaining at the conceptual level. The navigation of the data base would then consist of specifying what parameters would satisfy the demands of a particular analysis. In contrast to this is the construction of a retrieval which is dependent on the underlying physical organization, and requires an understanding of the system's execution logic. Such retrievals involve the user in the mechanics of interrelating physical structures and the inclusion of logic tests to account for their proper use. This aspect of data independence is common to all operations on the data base, not only user defined retrievals. This was suggested in the example of dumping a patient's data. This function should not require the user to manage the integration of the individual record types. Another example is found in batch data input. Additions and updates to the data base, while targeted for specific record types, need not be split up because of the demands of the physical organization of the data. The user should be able to apply a single source file to the data base and let the system manage the distribution to the correct record types.

B. SIR's Approach to Data Structuring and Data Retrieval

SIR is a case oriented data base management system. In SIR a case is a collection of data records of one or more types which are measured for a single
subject. For the CSSCD project, a patient conveniently fits this primary structuring unit. However, whether the case is an individual or a group (e.g., a family, or a state in a census analysis) would depend on the nature of one’s study. Each case is distinguished by a unique user supplied identifier, the case-id.

A SIR data record is a collection of individual data items identified by a record type number (and name, if desired). Since it is possible to have more than one occurrence of a record, additional identifiers may be needed for unique identification. SIR uses SORT IDs for this purpose. These identifiers, as well as the case-id, serve as the basis for establishing links between cases and records. These relationships and the control of the retrieval process are discussed below.

The “Schema Definition” is used to define all the attributes of a record type and the variables within it. This definition is the union of a variety of commands for specifying read/write security (at the record and variable level), variable and format lists, and missing-value definitions. SIR also includes in this definition integrity tests for individual variables and among variables, and calculations to be performed during record input. (These additional attributes are discussed further in Section II. — "Data Integrity".) A codebook can be generated with the full set of record and variable attributes and descriptors by invoking a system utility.

SIR provides a special record type for each case called the Common Information Record (CIR). During a retrieval, this record is automatically read for each case and is always available while the case is being processed. The record minimally contains a case-id and a count of the record occurrences in the data base for each record type within a case. The user may also place study variables in this record. The criteria for choosing variables for inclusion in the CIR will be evident in the use of this record during retrieval.

The “retrieval program” is the mechanism for user directed access to the data base. These programs can be used to interrogate the data base, display information, update data, perform calculations, and construct records to be processed by other procedures (e.g., a report generator, or the interface to statistical software packages). SIR provides a rather broad set of commands and built-in functions to manipulate data. This discussion of SIR examines those commands which can be used to navigate the data base, and the methods for making concrete the relationships established in the case and record identifiers.

SIR provides a set of block structured commands to control data base access. Consistent with the case as the primary structural unit, controlling case access is performed by the PROCESS CASE commands. The basic retrieval structure is thus to pass through the data base in case-id order. With additional specifications on this command, the user may restrict the cases accessed (by specifying a list or range of case-ids, or a limiting number of cases, or a random sample size. The list option can be used to great advantage by wisely coding the case-id. The CSSCD project significantly restricts retrievals by coding the clinic number as the first two characters of the case-id.

The functionality of the CIR becomes more apparent now. As noted, the CIR is automatically accessed for each case unless a limiting option of the PROCESS CASE command is specified. By placing variables within the CIR, the user eliminates the need for further record access within a case. Values of CIR variables can also be tested as the basis for further record searching. The variables selected for inclusion are normally those thematic to the study. They may also be variables which summarize large amounts of data on the data base, or are created as the result of complex or costly calculations which would have to be performed on a regular basis. The fact that a CIR is read for each case during a retrieval is important to understand in considering SIR. The user is most able to take advantage of SIR’s efficiency when the ratio of records to cases is high. The selection of a subset of cases, even when all CIR’s must be read, will result in many fewer records being accessed.

Linkage between cases, for sharing data or handling the problem of matched controls, is solved by a networking feature. A network model implies that there can be more than one path to a record or group of records. The normal SIR retrieval path of accessing cases in case-id order can be altered with the CASE IS command nested within the PROCESS CASE block. When the CASE IS block is completed, the system automatically returns to the case that initiated the branch. It is the user’s responsibility to store the case identifiers as data items in the cases to be linked. The CSSCD project maintains newborn and control ids to establish bidirectional links between patients and matched controls. We are also able to store family data under only one patient by
storing the selected patient's case-id with all the other patients in the family. A note of warning should be given in regard to this implementation. Care must be taken to insure that the case containing the single copy of the data is not deleted, or that the data is migrated to another patient prior to deletion of the other case.

The access of record types is controlled by another block structured command, PROCESS REC. This block implies a loop, accessing each occurrence of the record type specified within a case. The user may define one or more blocks in any order desired. Establishing links between record types is achieved by using an optional "control phrase" and nesting PROCESS REC blocks. The control phrase is used to specify which value or values of the SORT IDS will define the records to be accessed, and may be specified as a single value or range of values. In the CSSCSD project, we have a record type for each acute event, but a single record type for all follow-up examinations. By nesting PROCESS REC blocks for these two record types, we can select only those follow-up records which are associated with a single acute event. The control phrase may also be used to simply limit the records accessed for a single record type, without regard to any other record types.

The above commands allow the user to define the order and selection of cases and records, providing the context for the manipulation of variables for calculations, displays, or updates. The retrieval program is also the mechanism for creating the interface to statistical software packages, and executing other system procedures such as the report generator. These procedures are alike in that they all require records defined in a single data format. Retrievals are therefore used to organize data into "summary records" from one or more record types in the data base. The variables to be included in the summary record are specified in a MOVE VARS list. The user controls when the record is sent to the procedure by its placement in one or more PROCESS CASE or PROCESS REC blocks. The construction of the summary file may take on a variety of forms; data can be sent once for each case, or once for each occurrence of a record; data can be aggregated for a group of records and then sent to the procedure; or data can be distributed from one record into all occurrences of another record type.

SIR's combination of data structures and retrieval language has enabled the CSSCSD project to model the variety of relationships found in the study. Users should plan to reserve time for designing the implementation of their study to make best use of the system. We have found SIR's retrieval language to be easy to use, and its case oriented approach very helpful in managing research related problems. Although SIR does not provide a complete measure of data independence, our retrievals have proven to be consistently accurate and reliable, which I think is largely due to the degree of data independence that SIR does provide.

II. DATA INTEGRITY

The problem of data integrity concerns ensuring that data in the database is accurate at all times. Research studies are typically concerned with the correctness of individual data items and relationships among these items. The very process of data management also introduces integrity problems. Operations performed on variables, records, or files must be applied consistently. If a variable is to be recoded prior to permanent storage, it should be done with the same algorithm, and in all occurrences of that variable. The structure of the data base must be maintained in conformity with the logic of the study. As an example, if a case is dropped from the study, data in all records and files must be deleted for that case. The data must also be protected from damage caused by user error, software problems, and hardware failures. Such problems are caused by a variety of sins and catastrophes - incorrect algorithms, mistaken selection criteria, a new version of the DBMS containing bugs, and operating system failures.

The methods a DBMS employs or provides the user for solving data integrity problems are not of equal value. Methods which are or can be embedded within the system itself are more desirable because they are less susceptible to misuse and thus provide greater protection. The less one relies upon following a "standard set of procedures", which often need to be interpreted and may or may not be up to date, the fewer chances will there be for corrupting the data base. As a final comment, it is also worth considering the ways in which the DBMS alerts the user to error occurrences and to possible corruption of the data base.

In the following discussion I have tried to present the major features provided by SIR for dealing with data integrity. It is not an exhaustive list (I omitted, for example, automatic data type checking), but does cover the scope of alternatives and SIR's general approach to the problem.
The "Schema Definition" (described in Section I.) is a primary place to insure data integrity in SIR. This is achieved by allowing the user to extend the definition of data items to include integrity tests and by defining a set of operations as belonging to a record type. Once defined and stored in the data base, the system automatically performs the tests and calculations during batch data input and retrieval updates.

Valid values and ranges can be defined for a variable or list of variables. Variables failing such tests can optionally cause the record to be rejected or the value set to "undefined". Logical inconsistencies among variables in a single record type can be tested to screen records.

Calculations and recodes can be defined for variables within a single record type.

An error report is generated defining the errors encountered with a dump of the offending records. SIR can also be directed to write rejected records to an external file. This is a very convenient feature - all errors, collected in one file, can be corrected and reloaded without having to search and select from the original input file.

Edits involving data from different record types (or cases), or needing more sophisticated procedures (e.g., statistical techniques), must be performed in a separate retrieval program. Edits requiring report forms and messages more meaningful to certain end users than those generated by the system also would require execution of a retrieval program.

Separate commands for adding and updating records can be invoked, which can help avoid some irregular results caused by entry of incorrect identifiers. In "add" mode, records that are meant as new records will not replace old records (i.e., records with the same key). In "update" mode, records containing values meant to update records already on the data base will not be added to the file if the old record is not present.

Maintaining the logical structure of the study data is, as noted above, another integrity consideration. Deleting information from the data base is one source of destroying this structure. By having all record types integrated into a single data base, deletions at the case level are easily and automatically processed by SIR. Deletions at the record level, however, do present a problem. Where record types are hierarchically related, the deletion of a "superior" record occurrence normally implies the deletion of all "dependent" record occurrences. (In data base terminology, this prevents the occurrence of "orphans".) As an example, if a treatment record is deleted for a patient, any associated follow-up records should also be deleted. This avoids the problem of trying to explain away report results that show, for example, follow-up exams performed on 127 patients, when another report shows that there were only 126 patients treated. In SIR, hierarchical relationships are created through the use of the data and are not ensured by the system. The user is responsible for correctly processing deletions according to the study logic. (It might be noted that orphans can also be created by adding data as well, since the dependent record can be added prior to the superior record.)

Data base modification is a major source for introducing errors. The scope of possible errors is much too vast to be discussed here. I would like, though, to note some points related to one commonly performed modification, the redefinition of record structures, which would include format changes and the addition and deletion of variables.

In SIR, a redefinition of a record type initiates the modification process. Data from redefined record types are unavailable until the actual restructuring is performed. The restructuring is done automatically when the data is unloaded. The system takes the newly supplied definition to control the procedure so the user does not have to manage the dropping of variables. Newly defined variables are set to undefined for records already on the data base.

A report generated by the system during redefinition includes a list of variables that will be deleted and those that have changes in data type or length. Because SIR separates the processes of redefinition and restructuring, any errors found in the list can be corrected before a costly but worthless modification is run.

Only users with the highest level security clearance can cause a record to be locked.

Computations, recodes, and accept/reject tests which are newly defined are not performed during
Restructuring. The user must execute a separate retrieval program to perform these tests and calculations.

Recovering from damage to the data base requires the user to maintain a separate intact copy of the data base. Efficient unload/reload facilities are provided for this purpose. A "verify file" utility exists to check the integrity of the data base after abnormal termination. Actually, the system forces the user to execute this utility by making the data base inaccessible to other operations. A "journaling" feature is an additional method for data base protection. During batch data input or retrieval update, the system can be directed to create a journal file of all records that are entered, modified, or deleted. When necessary, the journal files can be applied successively to the last intact copy of the data base to reconstruct a complete data base.

Journaling is also an extremely useful tool for supporting and protecting on-line update. By using a journal file during a series of on-line update sessions, a system failure would, at most, cause the loss of work for the current session. The allocation of the journal file can be integrated into the on-line procedures themselves.

III. INTERFACE TO STATISTICAL SOFTWARE PACKAGES

While the decision to use a DBMS may solve a variety of data management problems, the resulting separation from the statistical analysis tools creates another set of significant problems. Before describing SIR's resolution of this situation, I think it is worth discussing some general issues involved in constructing an interface.

A. Problem areas

1. Integrity considerations

There are generally two methods for establishing an interface to statistical packages: by creating either a "system" file or an "external" file. An external file is normally easy to create, minimally requiring the user to define two variable and format lists - one to build the external file and one enabling the statistical package to read it. The creation and maintenance of two separate lists is a highly error prone process, as differences in the list can cause the loss of data or the incorrect assignment of data to variables. Data attributes such as missing value definitions, variable labels, and value labels are normally not part of an external file definition, so they must all be redone for use with the statistical package. This redefinition is also open to error.

The creation of a system file implies that the conversion and formatting of data is performed by the DBMS or by the user programming these functions as an extension of the DBMS. Integrity problems can be avoided if these functions are generalized and provide all the features required by the user. It should be pointed out that the data attributes within the data base are not easily accessible to users choosing to write their own interface program. Moreover, many DBMS do not use some of the attributes which are common to statistical packages, such as missing value indicators.

2. Integration of the interface within the DBMS

The ability to construct an interface file from a DBMS does not necessarily indicate the relative merit of the function. The manner in which the mechanism is integrated into the system must be considered. It is most advantageous if the interface is fully integrated into the retrieval system so that the user can benefit from all of the system's functionality. The assumption is that the user will be selecting a subset of the cases and/or data in the data base that this should be done at the data base level, not after the file is constructed. The user should also be able to create new variables during the retrieval and pass these along.

3. User time and effort

The construction of a fully documented, fully labelled, and accurate system file is a considerable investment in time, even for quite small files. The interaction between the DBMS and the statistical software is a frequent and ongoing process, not as a one time operation. While many passes from the DBMS to the statistical package will be identical (in terms of variables included, not necessarily cases selected) one must assume quite a variety of combinations; otherwise we probably wouldn't have chosen to use the DBMS in the first place. The user should be able to construct interface files as demanded by the needs of the study and not be constrained by the laborious process of redefinition. Not to be ignored is the displeasure of users who are required to redefine for the statistical package what has already been painstakingly done for the DBMS. Programming can be used to eliminate some of this work, but this,
too, implies a significant staff effort.

B. SIR’s interface to statistical software packages

SIR has facilities to create user formatted files as well as system files for two statistical packages—SPSS and BMDP. Currently SIR cannot directly create a SAS data set. Our procedure is to build an SPSS system file and then use the SAS CONVTS procedure to create the SAS data set.

Since SIR provides a fully integrated procedure for creating system files, we almost never use an external file to establish the interface. The rectangular file required by SAS procedures is constructed during a retrieval program as outlined above. The user controls the cases and records selected and defines the variables of interest in the data base. SIR handles all data conversion and formatting and automatically transfers missing value definitions and variables and value labels. In addition to the selection criteria, the user may apply all the standard retrieval commands for creating new variables through calculations and recodes. Missing value definitions can be given to new variables, and old variables can be given new definitions. Additional processing options can be invoked for the procedure such as sorting the summary records or selecting a random sample of the records.

We have encountered only a small number of problems with the SIR/SAS interface process:

Value labels are lost (see the CONVTS procedure in the SAS User’s Guide.)

JULIAN dates are handled differently in the two systems. SIR uses the number of days since October 15, 1582 (the beginning of the modern Gregorian calendar), while SAS uses January 1, 1960 as the reference point.

Blank/zero distinctions, if needed, should be handled prior to SPSS system file creation.

Missing value definitions are handled properly, but SIR maintains only three values.

Although we incur an extra expense in converting the intermediate SPSS file, the cost is incidental in comparison to the initial SIR retrieval and the analyses to be performed.

This combination of SIR and SAS has had some unforeseen but quite positive consequences. Our project, like many others, has a division of labor between a data management group (DMG) and an analysis group consisting of biometricians, epidemiologists, and statisticians. The DMG has the responsibility for maintaining the data base and creating the SAS data sets. The analysts do not directly use the DBMS. The detailed understanding and familiarity set the actual data base implementation, as well as the technical use of the DBMS, by the DMG is more likely to create an accurate and reliable data set. Analysts do not have to learn a new system, nor concern themselves with the specific implementation of the study. They can work with an immediately usable file within a familiar system. This also allows for greater and more orderly control over the data base, with less opportunity for introducing errors (the DMG is more than capable of generating its own).

Records which are meant to be accessed sequentially will normally be processed more efficiently in SAS than SIR given their different retrieval methods. Our construction of a SAS data set does, in fact, imply that we have selected and organized our data to be used as a sequential file. When using SAS on these subsets, we are able to take advantage of a more efficient processing mode, and reduce our expenses. This savings can be significant since these files are often reused for different analyses and can require multiple passes to perform the statistical procedures.

I think it is hard to overestimate the positive contribution to the research process that this automatic interface has given us. Our project felt it was necessary to use a DBMS like SIR to best meet the variety of its data management tasks. The ease of use and inherent reliability of SIR’s system file interface has made it possible to also meet our research requirements for statistical analysis.
FOOTNOTES

[1] The features discussed in this paper are found in Version 2.0 of SIR, which is available on these computers: IBM models under OS/VS (Batch), TSO and CMS (Interactive); Vax 11/780 under VMS; UNIVAC 90 series under VS/9; PRIME 450 through 750 under PRIMOS.

Version 1.1 of SIR is available on CDC 6000 and CYBER series computers under all operating systems. However, Version 1.1 does not have some of the facilities noted in this paper, and is a considerably less powerful system. Conversion to Version 2.0 is being investigated for these computers.

[2] A "record type" is defined as a collection of data items defined by a particular format. A "record occurrence" is the actual presence of data in the data base. There may be zero, one, or many record occurrences for any one record type. Other data base terminology could be used. The main idea is that the data should be physically and logically divisible into workable units.