11-1-1981

Using Database Management Software for a Stream Resource Information System

Lewis D. Hopkins
University of Illinois at Urbana-Champaign

Marc P. Armstrong
University of Illinois at Urbana-Champaign

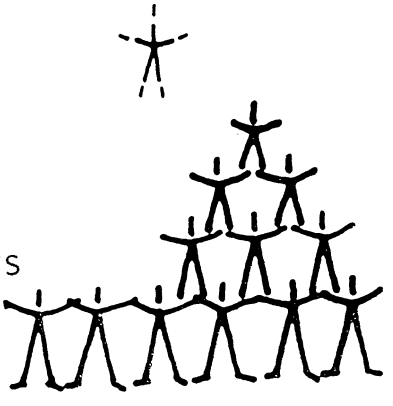
Geneva Belford
University of Illinois at Urbana-Champaign

Regional Landscape Planning

- LANDSCAPE PLANNING ISSUES & PROCEDURES
- LANDSCAPE VALUES, PERCEPTIONS & PUBLIC RESPONSES
- COMPUTER AIDED LANDSCAPE PLANNING

Proceedings of 3 Educational Sessions

1981 ANNUAL MEETING
AMERICAN SOCIETY OF LANDSCAPE ARCHITECTS



Organized by:
ASLA Task Force on Regional Landscape Planning

USING DATABASE MANAGEMENT SOFTWARE FOR A STREAM RESOURCE INFORMATION SYSTEM

Lewis D. Hopkins
Marc P. Armstrong
Department of Landscape Architecture

Geneva Belford
Department of Computer Science

University of Illinois, Urbana, Illinois, USA

Abstract. A few states now have land resource or stream resource information systems with software for relating data locations in terms of stream flow or watershed patterns. These information systems, however, typically rely on very simple data file structures, usually a coded value for each data item at each location. Using database management software provides the complex data structures needed for effective land and stream resource information systems and provides several additional benefits.

1. Introduction

Regional landscape planners have developed many databases, usually aimed at overall assessment of comprehensive plan alternatives (e.g., Fabos & Caswell, 1977; Steinitz, 1978). Land resource information systems intended for general use have also been developed. Deuker (1979) reported that existing land information systems were not used because hand methods were cheaper or data in the systems was not in a form or of a spatial resolution required for the tasks requested by users. Rather than merely add to the effort to increase data resolution or to the effort to collect additional data in machine readable form, the present project attacks the failure of information systems from a different direction: the use of powerful, off the shelf database management programs.

We are developing a stream resource information system under contract to the Department of Conservation of the State of Illinois. The major expected uses are to aid in evaluating more than 2,000 permit applications that the department must review each year and to provide analytical summaries of streams data as a basis for legislation and for department policy. The system may also be used in other department activities, such as acquisition planning, and impact analysis. A major characteristic of our effort is to focus our attention on the use of database management software and on the extensive experience with database management in industry and government (see e.g., Martin, 1977). The major function of database management software is to structure the storage of data to provide for efficient data retrieval for a wide range of types of retrievals, both foreseen and unforeseen. One advantage of using a database management software system is that it allows for much more complex data structures and retrievals than are possible with the traditional set of map-like files of data.

The intent of this paper is to describe briefly the issue of data structure in the design of the streams inventory and then to describe the use of a database management software package for the streams inventory project

as a way of illustrating the potential benefits of such software for use in land and stream resource information systems.

2. Data Structure

The crux of data management is to store data with minimal redundancy (duplicate storage of the same piece of data) while allowing for efficient retrieval with respect to a large range of data relationships. The problem in its simplest form is the difference between searching for a data entity in an ordered list versus searching for it in an unordered list. It is efficient, for example, to find a particular name in an alphabetically ordered list. If, however, the task were to find persons of a given age, it would be more efficient to have a list ordered by age. In order that both searches be efficient, the data would have to be stored in two lists, thus leading to redundancy. A good database management system allows the user to specify what data items should be used to order the storage of data and provides for the creation of apparent duplicate lists, where needed, by an efficient indexing mechanism. The user does not have to be concerned with the complexities of the actual physical arrangement of the data in storage.

2.1 Rectangular files

The simplest data structure is a "rectangular file" or table (see figure 1.) It is usually ordered on the values in the first column, which are identifiers of (or "keys" to) the entities being described by the rows of the table. In this structure, each entity must have the same set of data items associated with it; hence, each row has the same number of columns. In our application an entity might be a stream segment and be identified by a stream segment number. If efficient retrieval is necessary only by

ENTITIES	DATA ITEMS		
	STREAM NAME	TOWNSHIP/RANGE/SECTION	LENGTH
10-00-00/10			
10-00-00/20			
10-10-00/10			
10-10-10/10			
10-20-00/10			
10-30-00/10			
10-30-10/10			

Figure 1: Rectangular file

stream number and each stream has the same data items associated with it, this data organization may be quite reasonable. It is easily implemented without special software. Such simplifying assumptions cannot, however, be made for a state-wide streams inventory with many potential uses for the information.

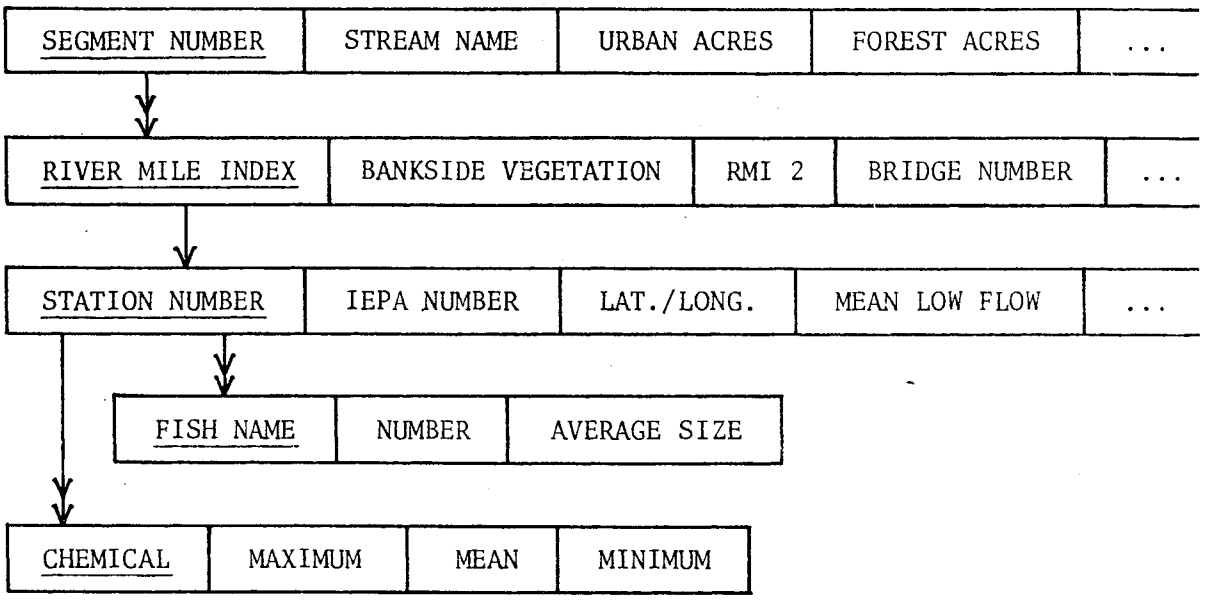


Figure 2: Hierarchical data structure

Notes to explain figures 2 & 3

- underlined words designate the 'key' variables by which data entities are ordered
- other words in a row of boxes are data items associated with the entity
- a single-headed arrow points to an entity, one instance of which is "owned" by and therefore readily accessible given an instance of the entity from which the arrow originates
- a two-headed arrow points to an entity, many instances of which may be "owned" by an instance of the entity from which the arrow originates

2.2 Hierarchical data structures

The diagram in Figure 2 shows a simple hierarchical data structure involving several kinds of entities. At the top of the hierarchy are stream segments. Each stream segment has a "row" of associated items of data that apply to the stream segment, much as a map grid cell has associated items of data. The row is a "record" of information associated with that stream segment. Each stream segment "owns" some number of River Mile Index (RMI) entities. Each RMI entity has a "row" of associated data items. Information about a stream segment can be found efficiently because the stream segment records are ordered by stream segment number; information about an RMI within a stream segment can be found efficiently because RMI records are ordered within stream segments. This data structure could be reduced to a table (imagine an outline with subheads). The benefit of a good database management system in this case is that it efficiently organizes the physical storage of the data even if many of the data locations in the table are empty. For example, there is no need to save space for an equal number of RMI rows after each segment number, even though there may be varying numbers of RMI entities per stream segment. Additional RMI entities could be added later and the physical arrangement of the data reorganized to fit the prescribed hierarchical data structure. Figure 2 shows a four level hierarchy. A River Mile Index may "own" a sample station and a sampling station may own fish and chemical records.

The hierarchical structure described above leads to more efficient storage than a rectangular file, but retrievals are still essentially via stream segment number. Additional types of retrievals will, however, make this data more useful for stream management and planning. First, it is desirable to retrieve efficiently data about hydrologically related stream segments (e.g., flow into, upstream of). Second, some users wish to retrieve data by the legal reference system of township/ range/ section rather than by stream segment. Third, some users wish to retrieve data by county. Additional desired forms of retrieval may arise.

The numbers that we are using as stream segment identifiers are designed so that most hydrologic relationships are implicit in the numbers themselves. The approach is similar to the approach in use in Minnesota (Thornton, 1981). Each stream is described by a hierarchical sequence of numbers, such as 10-30-20. The 10 indicates the final outflow stream (e.g., the Mississippi); 30 indicates the third tributary of stream 10; and 20 indicates the second tributary of stream 10-30. The zeros are used as place holders so that additional tributaries may be added or corrections made without disrupting the order. In addition to this stream number, each stream is divided into reaches, numbered from the mouth to the drainage divide. A stream number and a reach number together form a stream segment number, such as 10-30-20/20, that is used for ordering data in the database.

This numbering system causes the data to be arranged so that the hydrologically related stream segments will be close together in the logical storage structure and indexed in such a way that they can be readily found. For example, all segments of a given stream would be in order when the stream segment number is used in its entirety as a key. All reaches on tributaries of the stream 10-30 would be between 10-30-00/00 and 10-30-99/99, and all segments with numbers between these two numbers would be reaches on

tributaries of stream 10-30. Downstream hydrologic relationships are only slightly more complex in that they require knowing the reach at which each tributary meets each successive stream into which it flows. This data can be stored as a data item associated with each stream segment. Note that aggregating data by watersheds requires simply searching a hierarchical tree; the stream segment numbers are a many level hierarchy in themselves. Tracing a flow pattern downstream, however, requires associating data entities upward in the hierarchy after obtaining downstream segment data.

2.3 Hierarchical structure with inverted lists

Some types of retrieval require more than a hierarchical organization of the data. One of the data items associated with a stream segment is a series of township/range/sections through which the stream segment flows. To use these to retrieve data, however, would require looking at every stream segment record ("row") to find all those with a particular township/range/section. This would be equivalent to looking through an alphabetized list of names in order to find persons of a given age. Because it is expected that township range will be used frequently for retrievals, it is worthwhile to make a second list, an "inverted list," of stream segment numbers associated with township/range/sections. This inverted list is ordered by township/range/section rather than stream segment number, so that searching it is very efficient. Once it has been used to find an associated stream segment number, that number is then used to find all of the data associated with that location, without duplicate storage of all of the data items. Figure 3 illustrates a hierarchical data structure with inverted lists for township/range/section and for stream names.

Similar inverted lists can be made for sampling stations and counties. The stream name list is interesting in that there are many streams with the same name. In Illinois there are at least 21 Indian Creeks and 15 instances of several other names. Because the names are not unique identifiers, a user will have to look at a list of the downstream hierarchy by stream name to decide which Indian Creek is the one of interest. A good database management system will allow for the creation of additional inverted lists if they seem to be justified by the frequency of a particular type of retrieval.

2.4 Modification of the structure

Data items for which there is neither a hierarchical ordering (e.g., stream segments are hierarchically ordered) nor an inverted list can only be used as the reference for retrievals by exhaustive search among all the appropriate records containing that data item. To find all the stream segments draining more than 20 sq. miles, for example, would require looking at every stream segment record. The system is designed in expectation of such retrievals, but they will be used for summary purposes, such as supporting proposed legislation. These searches can, therefore, be slow and relatively costly because they will occur infrequently. If such a search were found to be needed frequently, an inverted list could be created to speed repeated retrievals. This open-ended flexibility is a major advantage of a good database management software package over traditional map-based systems. Such searches are also made exhaustively in map-based systems, but they do not provide for creating and storing a new inverted list. Map-based systems

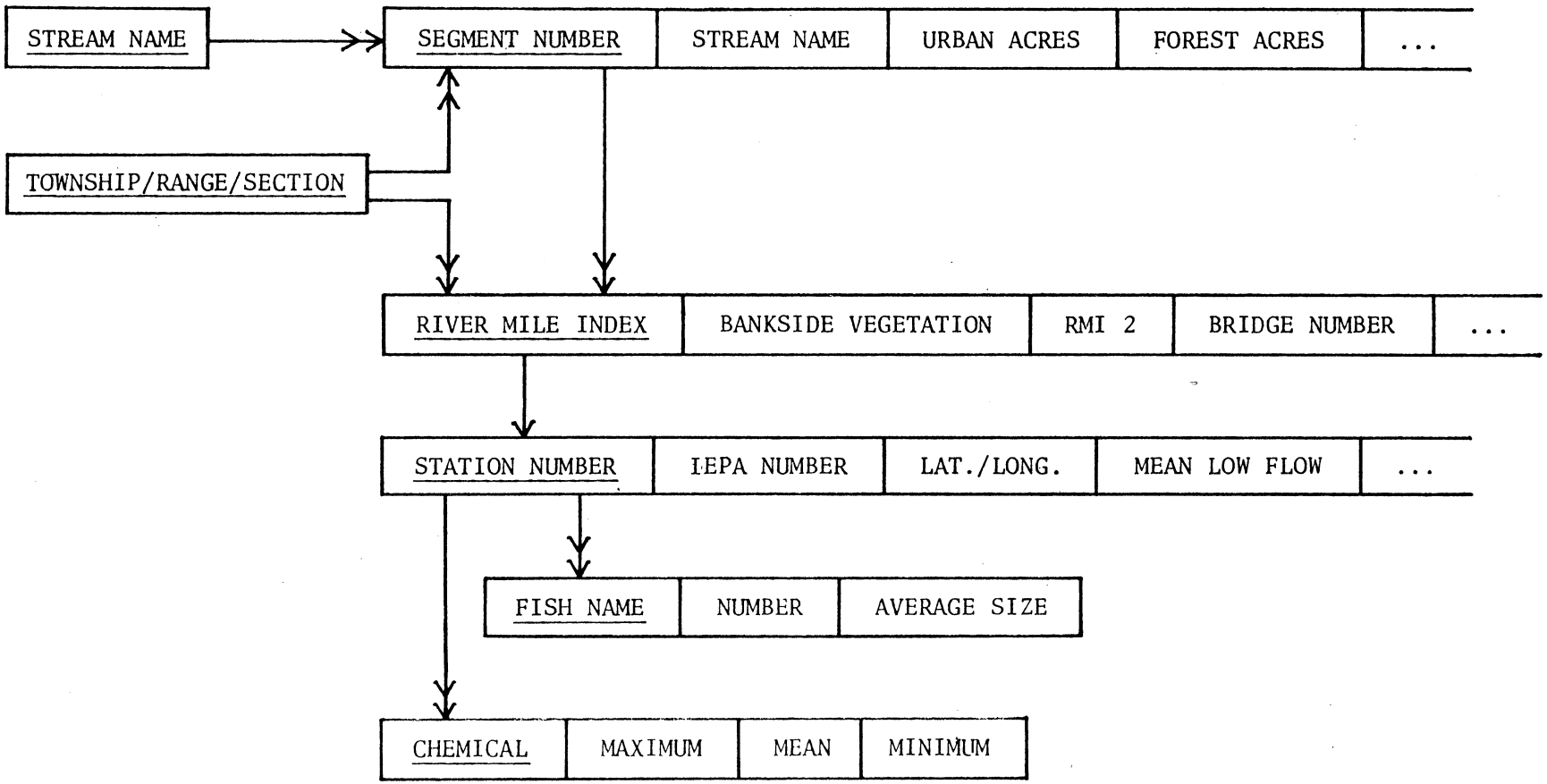


Figure 3: Hierarchical data structure with inverted lists

are usually ordered only by cell location in some coordinate system, or by some order of geographic areas in a polygonal map system. The creation of the hydrologic structure file in Minnesota was essentially the creation of an inverted list to allow the retrieval of data in the Minnesota Land Management Information System in terms of hydrologic relationships.

3. Use of Database Management Software

The database management software chosen for the Illinois Streams Inventory, Scientific Information Retrieval (SIR) (Robinson et al., 1979), was chosen in part because of local advantages; it was already available and being supported on the University of Illinois campus. Our description of specific tasks in terms of SIR should, therefore, not be construed to imply that SIR would be the database management software of choice in some other situation; it serves only to illustrate the characteristics of a suitable system. One way to describe the role of SIR in implementing an information system is that it is a high level language for storing and retrieving data (like SPSS, both in being a specialized high level language and in being very similar in actual syntax). The major tasks in implementing the information system are defining the data structure ("schema") and designing retrieval programs. First, explanations are given of the code to define part of the schema for the streams inventory. Then an example is given of a retrieval written in SIR. A user could learn to program retrievals directly in SIR, or could use retrievals prepackaged by the designers of the information system. The system can thus be very open-ended.

3.1 Schema definition

SIR is essentially a hierarchical database management system. (Multiple hierarchies are possible, however, which allow the description of somewhat more complex structures.) Our first task, then, was to decide what type of entity should form the highest level of the hierarchy. In SIR terminology, these entities are known as "cases". For each case multiple records, of various types, and themselves organized hierarchically, describe the various attributes of the entity. A record is a logical grouping of individual data items (variables). Given the nature of the data and the ultimately desired attributes of the system, a stream segment was chosen as the most natural entity to "own" many types of records of varying sizes. For example, a segment can own a single, short record containing physical characteristics--length, elevation at mouth--and several long records pertaining to sampling stations located in the segment.

Once the definition of a case is chosen, the exact structure of the database can be specified. In SIR, there are two general sets of specifications that must be given: case related and record related. The case related commands specify the estimated size of the database, providing information needed by the system to allocate space and set up internal indexes. The record commands specify the form of the data items. Other database management systems would require similar information, but in slightly different form.

3.1.1 Case related specifications

The first task is to define the CASE ID variable. This is the major selector variable in the database and it must appear on every record. Its effective use can greatly reduce retrieval times and costs. Using the stream segment number as the CASE ID variable allows each segment to be identified by its position in the stream network.

The next task is to estimate the total number of cases that will be included in the database. Also at this stage, the average number of records per case must be given. If no average can be calculated a reasonable upper bound can be specified. Upper limits must also be specified for 1) maximum number of record types, 2) maximum number of records per case, and 3) if input records are longer than the standard 80 column card image, the maximum number of input columns.

3.1.2 Record definition

Each record type in the database is assigned a number and a name--record type 3 might contain vegetation information. Within each record, variables can be specified as SORT IDs. Such variables provide for additional levels of hierarchical nesting of the database. Each record type must have a variable list and an input format, which together describe the input record for a single record type; each variable in the variable list takes up a prescribed number of columns on the record as given in the input format. There is a one-to-one correspondence between the variables and their order in the format statement.

As with most database management software, SIR allows the user to specify, either for a certain variable or a range of variables, values that are to be accepted as valid. Those values not meeting the conditions will be rejected. This checking is useful when it can be assumed that values other than acceptable ones are caused by coding or keypunching errors. For example, if there are 52 soil types in a study, each with a two digit code 01-52, any number greater than 52 would be rejected as incorrect, and reported to the user as being out of range.

SIR provides security of access to specified cases, record types, or individual data items. Security codes can be specified so that only certain users can access certain data. The location of archeological sites is an example of a data item that might be restricted. Security can also be specified so that only certain persons can change particular data in the system, thus protecting quality of the data.

Data manipulation features follow SPSS (Statistical Package for the Social Sciences) conventions closely. The COMPUTE, RECODE and IF statements are useful for creating new variables during database entry and retrieval phases. SIR also provides the capability to label variables and values within variables so that reports can easily be printed as verbal information even though data are stored in numerical codes.

3.2 Information retrieval

Information can be retrieved in SIR either by taking advantage of the inherent hierarchical organization or by selecting cases or records on the basis of values of particular data items. The following example considers all the cases (stream segments) upstream of and including stream segment 10-30-00/10; that is, it considers the entire watershed.

```
RETRIEVAL
FOR EACH CASE    LIST= '10300010' THRU '10309999'
FOR EACH REC     1
MOVE VAR LIST    LENGTH, DEVAC, SEGNUM
COMPUTE          ACIND= DEVAC/LENGTH
WRITE            SEGNUM,  ACIND
END
VAR LABEL        ACIND, DEVELOPED ACCESS SITES PER RIVER MILE/
PLOT             SCATTERGRAM= LENGTH WITH DEVAC/
                 TITLEX = 'NUMBER OF DEVELOPED SITES PER SEGMENT'/
                 TITLEY= 'LENGTH OF STREAM SEGMENT IN MILES'/
FINISH
```

In this example the length of each segment and the number of developed access sites in that segment are used to compute access sites per mile. This new variable is then given a label for reporting purposes. The stream segment number and the developed access sites per mile for each segment are printed. A plot is made of the number of access sites versus length of segment. Note that the report uses descriptive labels, not the short mnemonics used as variable names. Reasonably efficient retrievals of this kind are possible for any level in the data hierarchy, such as all river-mile indexes between two river miles within a stream segment.

The next example illustrates retrieval by specific properties described in the data items.

```
RETRIEVAL
FOR EACH CASE
SELECT CASE IF   (PUBSITE GE 1)
FOR EACH REC 1
MOVE VAR LIST    STRMNAM, PUBSITE, COUNTY 1, REACHNUM
WRITE            STRMNAM, REACHNUM, PUBSITE, COUNTY 1
FINISH
```

This code selects all cases (stream segments) that have one or more public access sites. These segments are then listed by stream name, reach number, number of public sites, and the county in which the outflow of the reach occurs.

These retrievals are intentionally very simple. In the last example, one might prefer to list the public access sites organized by county with the stream segment name. This retrieval is quite possible, but involves several additional lines of code because segments may flow through more than one county, or be the boundary between two counties. Retrievals can also be coded to create files for statistical analyses using SPSS, or analyses previously coded in FORTRAN.

4. Conclusions

A major concern of database management is to make data available in usable form to many different kinds of users who use the data in different ways. A good database management system will: 1) minimize redundancy in the data, but allow for many types of relationships in retrieving data; 2) provide for the logical data structure (the schema) to be independent of the actual physical storage of the data so that the user will be insulated from changes in hardware; 3) allow data items to be changed or added without the user having to be concerned about efficient reorganization of physical storage; 4) provide for unanticipated queries by providing a high level language for data retrieval; 5) provide report generating capabilities that allow for compact storage of data in numerical codes, but provide full English language descriptions of data in retrievals; and 6) handle housekeeping chores, such as checking for out-of-range data values, and being secure against inadvertent or unauthorized data changes or retrievals.

Use of a sophisticated off-the-shelf database management system, instead of attempting to write one's own software, provides these benefits with minimal effort. Whether our emphasis on such database management software, rather than on programs for analysis and mapping, will increase the usefulness of a land or stream information system can only be determined by implementation and use of the system.

References

Dueker, K., 1979. "Land Resource Information Systems: A Review of Fifteen Years Experience," Geo-Processing 1: 105-128.

Fabos, J., & S. Caswell, 1977. Composite Landscape Assessment, Research Bulletin No. 637, Massachusetts Agricultural Experiment Station, Amherst, Massachusetts.

Martin, J., 1977. Computer Data-Base Organization, 2nd edition. Prentice Hall, Englewood Cliffs, N.J.

Robinson, B., G. Anderson, E. Cohen, & W. Gazdzik, 1979. SIR-Scientific Information Retrieval User's Manual, SIR, Inc., Evanston, Illinois.

Steinitz, C., 1978. The Interaction between Urbanization and Land Quality and Quantity in Environmental Planning and Design, Landscape Architecture Research Office, Harvard University.

Thornton, J., 1981 "The Stream Inventory and Data Retrieval Systems Program," mimeo, Minnesota Department of Natural Resources, St. Paul, Minnesota.

Acknowledgements

The streams inventory referred to in this paper is being developed under a contract with the Illinois Department of Conservation.