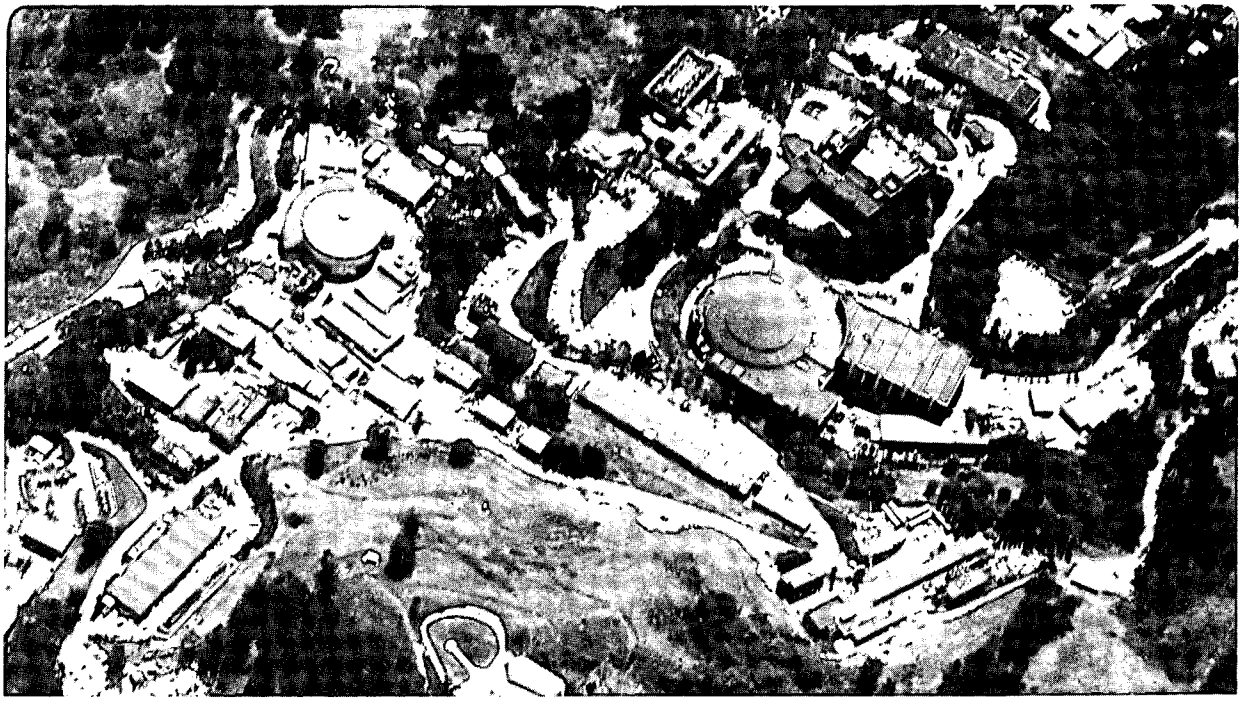
Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA, BERKELEY

Information and Computing Sciences Division

Use of the LBLSTAFF Database

A. Konrad

December 1987



Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

DISCLAIMER

PUB-3074

USE OF THE LBLSTAFF DATABASE

A. Konrad
Office of Computing Resources
Revised 16 December 1987

Purpose.

In planning the forthcoming ICS, the Project Team has reviewed how various telecommunications services are presently provided. One of those, directory assistance services, is presently supported by LBLSTAFF, a locally-implemented SPIRES database.

This paper describes how the present system provides directory assistance services, related applications, and those inherent features which might require ongoing support, whether in LBLSTAFF or a replacement product from an ICS vendor.

I. SPIRES at LBL.

In 1981, the Laboratory selected the Stanford Public Information Retrieval System (SPIRES) as one database system to meet LBL database needs, particularly those of the Libarary Department. For several years, the Library had been using SPIRES at SLAC for an online book acquisitions system and SPIRES at Stanford for shared cataloging. Selection of SPIRES was based on administrative, scientific as well as bibliographic needs.

Because LBL did not have a user-accessible IBM environment required to run SPIRES, agreement was reached with the Berkeley campus computing facility in Evans Hall to provide access to their system for LBL users and to run SPIRES. In 1984, the campus also purchased a SPIRES license and now separate campus and LBL SPIRES environments are supported on the same machine.

A new version of SPIRES is distributed annually by the SPIRES Consortium, an organization consisting of Stanford University and about 50 universities including Berkeley, UCSF, CUNY, Cornell, Harvard, Princeton, and Yale, as well as NASA Goddard, Max Planck Institut für Astrophysik, and the National Center for Atmospheric Research. The SPIRES Consortium also provides consulting support by telephone and over BITNET. The annual membership fee is $7,000, which covers both software maintenance and systems consulting.

SPIRES runs on the Berkeley campus IBM 3090 dual processor under VM/CMS. Access for LBL users is at 9600 baud via Develcon to one of several IBM Series/1's, which provide terminal emulation for ASCII terminals.

The Laboratory has in excess of 30 SPIRES accounts on the campus system.

Major SPIRES users at LBL

Database	User	Subject
LBLSTAFF	Various	LBL employee and guest information
DRAWINGS	Mech. Eng	200,000 Mechanical Eng. drawings
ENGNOTE	Mech. Eng	Mechancial Engineering Notes
MIST	CSR	10-year Materials Science Database Project, eventually to run at Sandia Laboratory in Albuquerque, to provide materials data nationwide.
ATTENDEES	Conf. Coord.	Several interrelated databases for conference management.
AWARDS	Directors Off.	Non-LBL awards for which Laboratory staff are encouraged to apply.
SOLAR	90 Library	Bibliographic collection in 90 Library
ADDRESSEES	SSC	SSC Public Information System
RPM *	Admin. Div. Off.	RPM Distribution
TRAINING *	Eng. Div. Off.	Courses and Students databases for LBL Safety Courses and for Emergency Preparedness Management.
LBLRIS	TID	LBL Reports Issued
LBLSER	LIBRARY	Serials subscriptions and control
ORDAC	LIBRARY	Acquisitions (SPIRES at SLACVM)
ADP	OCR	DOE/GSA ADPE Inventory
STATIONS *	CCR	ICS Telephone Survey
BLDG	anyone	Building number, name, manager, ICSnode

* Logically linked to LBLSTAFF for employee/guest data.

II. Description of SPIRES Functionality.

A database management system is a collection of software tools to provide relatively easy processing (entry, storage, retrieval, and presentation) of like data. The tools that comprise SPIRES are suitable for general applications, rather than limited to a particular one such as in "canned" applications software. SPIRES is particularly well suited to bibliographic applications, is appropriate for much scientific data, and has been used for several large administrative programs as well.

SPIRES functionality includes:

- Inverted lists as indexes for efficient searching
- Index record-types can serve as goal records for multiple views of data
- Versatile query language
- Multiple elements passed to a single index, element values passed to several indices
- Virtual indexes and indirect searching
- Textual, non-columnar data accommodated
- Variable length elements
- Optionally-occurring elements
- Multiply-occurring elements
- Structures ("repeating groups") to retain logical associations
- Variety of data element types
- Personal name and date processing
- Case handled intelligently
- Word indexes
- Protocols language tools
- System procedures and functions; system variables
- Ease of adding, updating, removing records
- English-like syntax
- Deferred queue maintained to retain all versions of a record for some period, enabling the user to "back up" and reference any of them
- Concurrent updating control for simultaneous use of database
- Security at the file, record, and element levels
- Automatic recovery in the event of a system crash
- Easy and fast to restore service in the event of a disk crash
- Versatile report generators
- Large capacity
- Inexpensive

These are some of the features for which SPIRES was originally selected and why usage continues to grow despite its inability to run on any LBL-owned computing system. SPIRES is ideal for a research environment because it does not require a central database administrator. Database owners can create, destroy, and modify their database definitions and data at will.

Section IV describes how the attributes above are used in LBLSTAFF and why they are important.

III. Description of the LBLSTAFF Database

Many LBL offices and departments maintain mailing lists that include LBL employees and guests. It follows that many of these lists maintain employee-guest data in duplication of one another. Further, none are likely to have manpower dedicated to tracking changes in employee data, such as mailstop, name, and department changes on a regular basis, but instead implement changes only when reported by the employee himself and only to staff supporting that particular database. Thus, the version of any particular employee's record in one database might be out-of-date, and also be inconsistent with that employee's record in another database. Maintenance of this data in multiplicate hill-wide is time-consuming inefficient use of Laboratory effort and not cost-effective. LBLSTAFF provides this service in a cost-effective way.

The LBLSTAFF database is comprised of 15 (maximum of 81 allowed) "record-types" (See SEC IV.1) or collections of records. Each record-type is comprised of like-records, each with a unique key and one or more other data elements. For example, a record in the employee record-type might appear:

KEY = 123
NAME = JOHN JONES
MAILSTOP = 50-212
PAN = 9876
LOCATION
  BLDG = 50
  ROOM = 215
  EXT = 5555

Of the 15 record-types in LBLSTAFF, the first (REC01) is comprised of employee records like the above example, and is considered a "goal record" record-type because employee data records are the goal of a search. The other 14 are index records used as an efficient means of locating employee goal records (Sec IV.1).

The LBLSTAFF database uses the GOAL-INDEX concept (Sec IV.2) extensively. LBLSTAFF is comprised of the following record-types. Those with subfile access are indicated as are the names by which subfile access is provided:

RECTYPE	GOAL RECORD	INDEX TO	SUBFILE ACCESS
REC01	EMPLOYEE data	--	LBLSTAFF SERVICE PHONEBOOK OCTOPUS
REC02	MAILSTOP	REC01	MAILSTOP
REC03	EMPLOYEE NAME	REC01	
REC04	PUBLISH FLAG	REC01	
REC05	--		
REC06	BUILDINGS	REC01	BLDG
REC07	EXTENSION NOs.	REC01	ICS
REC08	PAYROLL ACCT	REC01	PAN
REC09	DISU	REC01	
REC10	CLASS	REC01	
REC11	ZIP CODE	REC01	
REC12	LEVEL-III	REC01	
REC13	LEVEL-IV	REC01	
REC14	ELECTRON. MAIL	REC01	
REC15	CODE	REC01	

Note that there are four subfile names for REC01. Depending on which subfile name is selected, some elements in REC01 may or may not be displayed, or special processing may or may not occur, as discussed below.

As of 10 December 1986, LBLSTAFF REC01 consisted of 11,449 employee and guest records, and an average of 600-800 online queries per day.

Because the LBLSTAFF file is constantly updated with name, building & room, mailstop, extension, payroll account number and other changes, it is an ideal source from which to publish the alphabetical portion of the LBL printed phone directory. A SPIRES protocol containing the necessary command language enables any staff member to produce a *troff* input file covering any specified set of PANs, or the entire "active" Laboratory staff. The protocol is invoked simply by entering the command PHONBOOK, answering the questions that are asked, and then forwarding the file to the LBL phototypesetter over BITNET.

The centralized maintenance of LBLSTAFF also makes it a desirable "public" resource for mailing labels, reports, and data in 'satellite' databases. LBLSTAFF enables staff supporting these adjunct applications to avoid redundant maintenance of the data already kept up-to-date in LBLSTAFF.

The LBLSTAFF file consists of several physical files and several logical subfiles. LBLSTAFF refers to the *file*, i.e., the whole collection of physical and logical entities. Where the *subfile* of the same name is referred to, the term "LBLSTAFF Subfile" will be used.

Data elements defined for REC01 in LBLSTAFF include:

Employee-ID (key)
Employee-name
Alt.name
Office-name

Mailstop
Location
- Bldg
- Room
- EXT

Service-notes
- Note
- Note date

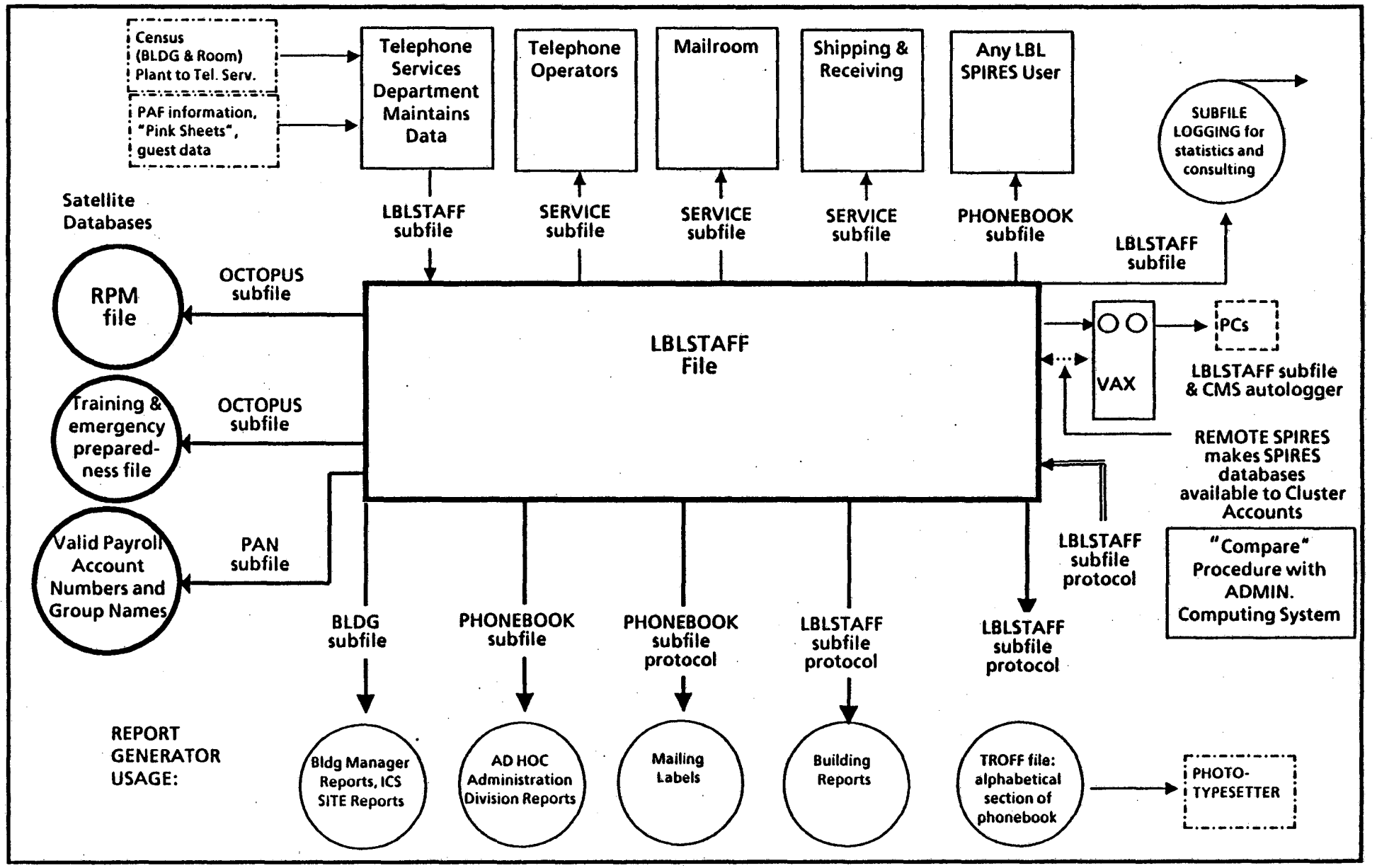
PAN
Class
Tenure
- Terminatation date
- Hire date

Basis
Publish
Home
- Home-priv
- Home Address
- City
- State
- Zip

Home-phone
- phone-priv
- area code
- home phone

Better-half
- spouse-priv
- spouse
- married

Date-changed
Notes
Level III
Level IV
Sex
Ethnicity
Computed version of EPO
Electronic Post Office address
L2
Sequence Name (virtual)


Census (BLDG & Room) Plant to Tel. Serv.
PAF information, "Pink Sheets", guest data
Telephone Services Department Maintains Data
Telephone Operators
Mailroom
Shipping & Receiving
Any LBL SPIRES User
SUBFILE LOGGING for statistics and consulting
Satellite Databases
LBLSTAFF subfile
SERVICE subfile
SERVICE subfile
SERVICE subfile
PHONEBOOK subfile
LBLSTAFF subfile
RPM file
OCTOPUS subfile
LBLSTAFF File
PCs
VAX
LBLSTAFF subfile & CMS autologger
Training & emergency prepared-ness file
OCTOPUS subfile
REMOTE SPIRES makes SPIRES databases available to Cluster Accounts
Valid Payroll Account Numbers and Group Names
PAN subfile
LBLSTAFF subfile protocol
"Compare" Procedure with ADMIN. Computing System
BLDG subfile
PHONEBOOK subfile
PHONEBOOK subfile protocol
LBLSTAFF subfile protocol
LBLSTAFF subfile protocol
REPORT GENERATOR USAGE:
Bldg Manager Reports, ICS SITE Reports
AD HOC Administration Division Reports
Mailing Labels
Building Reports
TROFF file: alphabetical section of phonebook
PHOTO-TYPESETTER

Usage of Subfiles in theLBLSTAFF File.

LBLSTAFF Subfile.

The LBLSTAFF subfile access is used primarily for adding, updating and deleting employee and guest records. Those accounts having permission to SELect LBLSTAFF generally have permission to see, search, and change the values of most data elements defined for REC01. Telephone Services staff perform virtually all adds and updates. To retain historical data, employee and guest records are not generally removed from the database unless an employee number changes. Only the following accounts can SELect LBLSTAFF: TPHHH (Telephone Services data maintenance staff), ADHDVN and SSDVN (D. Neilsen), and KONRAD.

SERVICE Subfile.

When the SERVICE subfile is selected, a format is automatically set to display employee records in a brief 2-3 line display, and then a SPIRES protocol automatically initiates to prompt to the user to enter a few characters of an employee or guest name followed by a carriage return. The screen clears, then forwarding and routing information is rapidly displayed, followed by a prompt for the next search. The prompt includes the time of day from the IBM 3090 system clock. (!)

The SERVICE subfile is used primarily by INFORTPH (Telephone Services attendants), MAILR (Mailroom staff), and SSRHP (Shipping & Receiving, BLDG 901). It is used for searching only.

PHONEBOOK Subfile.

The PHONEBOOK subfile is accessible to anyone who can use LBL SPIRES. It displays only those data elements that are displayed in the printed LBL telephone directory. For example, PAN is displayed, termination date is not. No adding, updating, or removing of records is permitted via PHONEBOOK access. Only searching and displaying records is allowed.

OCTOPUS Subfile.

As a cost-effective alternative to Laboratory-wide redundant maintenance of personnel data, LBLSTAFF provides subfile access via the OCTOPUS subfile for use by mailing list, training, and other applications. When Telephone Services updates an LBLSTAFF record, the new information is automatically and immediately available to other users when the record is displayed via a satellite database. Users need only maintain the employee or guest ID's in their satellite subfile records which acts as pointers to fetch the corresponding records in LBLSTAFF (OCTOPUS). This functionality is provided using virtual redefined elements and phantom structures. For example, an RPM record is stored:

RPM-ID = 47
EMPLOYEE-ID = E490429

but appears;

RPM-ID = 47
EMPLOYEE-ID = E490429
NAME = Allan Konrad
PAN = 9191
Mailstop = 50B-2258
Location
    BLDG = 50B
    Room = 2258C
    EXT = 5458
etc.

The OCTOPUS subfile is accessible to several accounts that are "owners" of SPIRES databases, including KOEHN (for the Administration Division's RPM database) and SEALY82 (for the Safety Training and Emergency Preparedness Management database). No adding, updating, or removing of records is permitted via OCTOPUS access. Only searching and displaying records is allowed.

MAILSTOP Subfile.

The MAILSTOP subfile has, as the key of its goal records, mailstop values from REC01 records. Usually, it is a building and room, but it need not be. Livermore style mailstops are also usable and prevent having to change a mailstop for an entire group when the group changes their physical location. Elements defined for mailstop records are:

MAILSTOP (key)
Building
Room
Payroll account number
Office-Name
POINTER

The primary purpose of subfile access to the MAILSTOP record-type is to add, update, and delete MAILSTOP records in order to validate mailstop values entered into employee records. That is, the MAILSTOP subfile acts as a lookup table to LBLSTAFF records to reduce the likelihood of an illegal mailstop from being entered into an employee record. Of course, the MAILSTOP subfile is also an index to REC01 goal records when the LBLSTAFF subfile is selected.

Only accounts which have permission to add and update employee and guest records can add, delete, or change MAILSTOP records. A SPIRES protocol is automatically invoked when the MAILSTOP subfile is SELected to assist in assuring that no MAILSTOP record is deleted so long as it contains any occurrences of POINTER, i.e., so long as an employee record has that mailstop as the value of its mailstop data element.

BLDG Subfile.

The BLDG subfile consists of:

BLDG (key)
BLDGNAME
BLDGABBRV
ICSSITE
BLDGMGR (Building Manager ID)
Virtual: Bldg Manager name, ext, mailstop, payroll account number, termination date

The BLDG subfile acts as an index to the LBLSTAFF goal records, as a lookup and translation table to convert building numbers into building names, and as a goal record in its own right. The translation enables publishing campus building names in the printed LBL directory maintaining them in employee and guest records. The lookup function also provides consistency to the appearance of converted building numbers. BLDGNAME is accessible to REC01 records as a virtual element. The BLDG subfile is also used to associate extension numbers with their ICS SITE.

Clay Sealy, the Laboratory Emergency Preparedness Officer, now maintains the Building Manager element in the BLDG subfile.

Access to the BLDG subfile is the same as for the LBLSTAFF subfile.

ICS Subfile.

The ICS subfile is goal record access to the extension number index for LBLSTAFF employee and guest records. It consists of:

EXT (key)
POINTER
Active structure
    ACTIVE (yes or no)
    Date effective
TIP ID or universal receptacle
PLACEMENT
NOTE

When ICS is SELected, it can access the employee subfile to associate employee names in terms of extensions, and the BLDG subfile to associate ICS site nodes in terms of extensions. When LBLSTAFF is SELected, it can provide the same data in terms of employee records. This provides "A in terms of B, and B in terms of A" functionality. The extension number index is maintained by virtue of Telephone Services staff effort to reflect telephone service orders into the LBLSTAFF (and, if needed, ICS) subfiles. The ICS subfile is accessible by the KONRAD account.

PAN Subifle.

The payroll account number (PAN) subfile is used much like the BLDG subfile, to provide a lookup translation table so that payroll account numbers in employee records can be translated into text descriptions the department or group. PANNAMES are maintained by Telephone Services. Virtual redefined elements in REC01 are used to provide the translated PAN to employee records. The PAN subfile also acts as an index to LBLSTAFF goal records when the LBLSTAFF, PHONEBOOK, or OCTOPUS subfiles are selected. PAN subfile elements are:

PAN (key)
POINTER
PANNAME

Access to the PAN subfile is the same as for the LBLSTAFF subfile.

IV. Examples: Why SPIRES-like functionality is Useful in Directory Services.

A brief description of some basic SPIRES concepts is presented to provide a better understanding of how LBLSTAFF fulfills the requirements of a directory services system and for its integration with related applications. The order of presentation is as listed in Section II.

1. Inverted Lists as Indexes for Efficient Searching.

The primary structural concept in SPIRES is use of multiple record-types as indexes to a goal record record-type. A record-type is a collection of goal or index records. A record is a collection of data elements and their values. The first element in each record is a unique key. Goal records are often contained in a record-type (RECTYPE) named REC01. Index records are in separate record-types, e.g., REC02, REC03, etc., though they may have any legal name. A simple goal record might appear:

KEY = 123
NAME = TOM SMITH
EXT = 5555

The characteristic that differentiates goal records from index records is that index records include an element called POINTER:

KEY = SMITH, TOM
POINTER = 123 (actually a hexidecimal address for "123" where 123 is the key of a goal record for Tom Smith)

The POINTER value is derived from the location of the goal record to which it points, much as the page number in the index in the back of a book points to the page on which a given topic is discussed. However in SPIRES, the user is not aware of the POINTER value. SPIRES uses it internally to present retrieved records to the user. For example,

Goal record rectype:

RECTYPE REC01
KEY = 1
NAME = TOM SMITH
EXT = 5555

KEY = 2
NAME = JOHN JONES
EXT = 4444

Index record rectypes:

RECTYPE REC02 (name index)
KEY = JONES, JOHN
POINTER = 2

KEY = SMITH, TOM
POINTER = 1

RECTYPE REC03 (EXT index)
KEY = 4444
POINTER = 2

KEY = 5555
POINTER = 1

Before doing a search in SPIRES, the user first SELects the subfile to identify which set of goal records are sought. A single goal record record-type may be accessed by several different subfile names. This is useful to control which users can see or update particular data elements and perform other tasks.

The FIND command in SPIRES searches the index records to locate pointers to goal records that meet the search criteria, and then reports the result to the user:

```
User:    Find ext 4444
SPIRES:  RESULT 1 RECORD
User:    Type
SPIRES:  KEY = 2
         NAME = JOHN JONES
         EXT = 4444
```

When a search is initiated, using the FIND command, the index record-type is searched, not the goal records. This is important in directory service as it speeds the searching process significantly in a large database, reducing waiting time to realtime users such as telephone attendants and mailsorters.

Obviously, the element POINTER can be multiply-occurring if there are several employees with the same telephone number:

Goal record rectype:

```
RECTYPE REC01
     KEY = 1
     NAME = TOM SMITH
     EXT = 5555
     EXT = 4444

     KEY = 2
     NAME = JOHN JONES
     EXT = 4444
```

Index record rectypes:

```
RECTYPE REC02 (name index)
     KEY = JONES, JOHN
     POINTER = 2

     KEY = SMITH, TOM
     POINTER = 1

RECTYPE REC03 (EXT index)
     KEY = 4444
     POINTER = 1
     POINTER = 2

     KEY = 5555
     POINTER = 1
```

Note that in all record-types, SPIRES stores records in order by their key. Thus, an inverted list, which serves as an index, is presorted, as are the goal records, conserving CPU resources during searching. The presorted aspect of inverted lists can be used by SPIRES to "drive" the order of output of goal records rather than performing the sort in realtime for mailing labels, building lists, etc.

Searching for records based upon the non-indexed elements is performed directly (sequentially) rather than via an index, which is equivalent to searching for a topic in a book page-by-page rather than using the index to go directly to the desired pages.

"A in terms of B, B in terms o f A" capability is provided merely by indexing. For example, suppose telephone set records contained a multiply-occurring element FEATURES, which is indexed. Displaying any single record will display all the features associated with that telephone set. The search FIND FEATURE = X retrieves all the records of telephone sets with that specific feature. One can display all the features of an extension, and all the extensions with a particular feature. This capability is used in LBLSTAFF for all elements that are indexed, e.g., PAN, and EXT.

The following elements are indexed in the LBLSTAFF (PHONEBOOK, OCTOPUS) subfile:

Mailstop
Employee name and
alternate name (same index)
Publish Flag
Building
EXT
PAN
Code
Display-Under
Classification (data not maintained)
ZIP Code (home addresses)
Level-III Distribution
Level-IV Distribution
Electronic Mail Addresses (data not yet acquired or entered)

2. Index Record-Types Can Serve as Goal Records for Multiple Views of Data.

Records in index record-types have the same structure as goal records, i.e., an element that is the unique key, followed other elements. The POINTER need not be the only other element besides the key in an index record. For example, in the EXT index record-type (REC03), another element indicating the color of the telephone might be defined. The record-type structure would appear:

```
REC03
    KEY
    POINTER
    COLOR
```

In the example,

Goal record rectype:

```
RECTYPE REC01
    KEY = 1
    NAME = TOM SMITH
    EXT = 5555
    EXT = 4444

    KEY = 2
    NAME = JOHN JONES
    EXT = 4444
```

Index record rectypes:

```
RECTYPE REC02 (name index)
    KEY = JONES, JOHN
    POINTER = 2

    KEY = SMITH, TOM
    POINTER = 1

RECTYPE REC03 (EXT index)
    KEY = 4444
    POINTER = 1
    POINTER = 2
    COLOR = RED

    KEY = 5555
    POINTER = 1
    COLOR = BLUE
```

In such a case, it may be desirable to process REC03 records in two different ways: as index records, and also as goal records in their own right. Such a record-type is called as a GOAL-INDEX record-type. Element values in a goal-index record, e.g. for the element color, can also be indexed for searching. Such a COLOR index might be REC04.

SPIRES provides access to goal records in goal and goal-index record-types as "subfiles", what users think of as "databases". Thus, in the example above, REC01 might be called the EMPLOYEE subfile and REC03 might be the PHONE subfile.

This dual role of record-types is one way that integrated non-duplication of data is achieved in SPIRES. For example, PAN and PANNAME elements do not both have to be stored in each employee record, only the PAN. And if the PANNAME is changed, it need only be changed in the PAN subfile which automatically causes the change to be reflected in employee records. Satellite databases such as RPM and TRAINING are important examples of how data can be stored and maintained in a single place, but be used as if it were stored in the many databases in which it is needed.

This dual role record-type also provides some of the relational capability in SPIRES. Much attention has been given recently to relational database systems which can provide useful on-the-fly views of data. However, practical considerations can remove some of the advantages, e.g.,:

- Multiply occurring elements are not usually accomodated in relational systems; rather, separate records are created, either needlessly duplicating data or else adding complexity to the file structure.
- Hierarchical structures are often not allowed. Structures must be normalized, imposing a flat file structure, proliferating records which must be recombined in some fashion, but most importantly, potentially losing the important meta-information provided by the organization of the data into a structure;
- JOINS must be computed at the time of search (usually prime interactive time), rather than overnight or as each record is added or updated, as in SPIRES. Relational systems are often CPU-intensive at search time for this reason, putting the burden of waiting on the searcher. Searching is generally much faster in SPIRES, though JOINS can provide better flexibility.
- It is sometimes difficult to accomodate non-tabular data in relational systems, e.g., bibliographic abstracts and personal names.

3. Versatile Query Language.

As indicated, searching for goal records in SPIRES can be done either using indices or searching the records themselves sequentially. Relational operators can be used in queries (=, >, <, ~=, >=, <=), range operators (from - to, between-and), content operators (HAVING, PREFIX, STRING, WORD, WITH, etc), and logical operators (AND, OR, NOT). For example,

FIND PAN >= 9000 and <= 9010

FIND MS HAVING 50

Compound searches are permitted, i.e., searching two indices:

FIND NAME SMITH AND PAN 9044

Inclusion and exclusion lists are recognized in queries using SEARCHPROCS and PASSPROCS (Section IV.15.1). For example, PASSPROCs and SEARCHPROCs for the ELECTRONIC MAIL element might specify the word AT and the symbol @ in an exclusion list.

Search results can be resequenced using the SEQuence command.

Search commands can use truncated values, e.g.,

FIND NAME G. SMI#

will find Greg Smith.

The capability of successive refinement of search results is allowed.

4. Multiple Elements Passed to Single Index, Element Values Passed to Several Indices.

A record in the LBLSTAFF subfile for a staff member with a newly-changed name might appear.

ID = 123
NAME = Susan Jones
ALTERNATE NAME = Susan Smith
etc.

Either search command

FIND NAME S SMITH or

FIND NAME S JONES or

FIND NAME S SMITH OR S JONES

will find her record, even though SMITH is the value of the ALTERNATE NAME element, not the NAME element, because both the NAME element value and the ALTNAME values are passed to the name index. This is important in directory services where a telephone or Mailroom attendant does not know the employees new surname or even that the employee has a new surname.

At present, no data element values are passed to more than one index. But, for example, it might be useful to pass LEVEL-IV values to both LEVEL-III and LEVEL-IV indices so that a search on LEVEL-III includes those employees on the LEVEL-IV distribution.

5. Virtual Indices and Indirect Searching.

It is occasionally useful to search for goal records based upon data that does not exist, and to have the records in the search result processed and displayed as usual. SPIRES provides two ways to accomplish this task, both by applying rules rather than utilizing stored data. The first is to build an index based upon data that does not exist, and to search that index in the normal way. For example, one might have employee records which contain a data element named ELECTRONIC.MAIL, where the values contained in the element are electronic mail addresses in the form 'KONRAD at UCBCMSA' or 'AKONRAD@LBL'. Such values are probably passed to an electronic mail index (say, EM), which is likely to be a word index. Prior to passing values to the index, @'s are changed to blanks, 'AT' is changed to null (to conserve storage), then the value is broken on blanks and each word passed to the index separately. This would enable the following searches to occur successfully:

```
FIND EM KONRAD
FIND EM UCBCMSA
FIND EM KONRAD UCBCMSA
FIND EM UCBCMSA KONRAD
FIND EM KONRAD AT UCBCMSA
FIND EM AKONRAD@LBL
FIND EM AKONRAD
FIND EM LBL
```

But it might be desirable to search on network names such as ARPANET or BITNET. But, such values are not stored in the ELECTRONIC MAIL element in employee records, nor need they be. A second index can be defined called NETWORK. Values of the EM element are passed to this index just as they are to the EM index, except that everything before the last word delimited by blanks is changed to null, and the remaining string is converted based upon a table, e.g., UCBCMSA is converted to BITNET, LBL is converted to ARPANET, or ARPANET AND BITNET, etc. Thus the following searches become legal:

```
FIND NETWORK BITNET
FIND NETWORK BITNET AND EM KONRAD
FIND NETWORK ARPA AND EM LBL
```

Non-existent data can be indexed and searched in other ways, such as from virtual elements which fetch data from other databases.

A second method, used by LBLSTAFF satellite databases, is to search an index that does not exist (a virtual index). This is called *indirect searching*. As mentioned, the RPM and TRAINING & EMERGENCY PREPAREDNESS subfiles contain no employee data other than the employee-guest ID which is also the key of the corresponding record in LBLSTAFF. When an RPM or TRAINING record is displayed, that value is used to fetch elements from the appropriate employee record and display them as if they were part of the RPM or TRAINING record. But what about searching? It would be useful to be able to search for employee data even though none is stored in such databases. For example, it would be useful to search a PAN index in the satellite subfile:

```
SELect RPM (or TRAINING)
FINd PAN 9191
```

even though no PAN element or index is stored.

In fact, this command sequence is legal and generates the correct search result because "indirect search" is implemented for the selected database. Indirect search operates as follows:

The subfile having been selected, the user issues a FIND command, as above.

SPIRES performs the search on a secondary database, in this case, LBLSTAFF, e.g., finds all the records with PAN = 9191.

Rather than reporting this intermediate search result, SPIRES then filters the search result to determine which records correspond to records in the selected (satellite) subfile (RPM or TRAINING) that contains the specified element-in-common, in this case, employee ID.

This filtered result is then returned to the user, who may then process (e.g., display, resequence) the records as with any search result. This has several advantages. First, less storage space is required. Second, only one index need be maintained whenever the PAN in an employee record is changed.

6. Non-Tabular Data Accommodated.

Many database applications employ data that is not easily or appropriately presented in a table. This is true from a formatting standpoint, where physically arranging such data into a table is inappropriate and difficult, e.g., the abstract of a scientific paper. For example, wrapping long text values on several lines so that words are recognized, starting a newline at the beginning of a word rather than in the middle of a word, makes generating legible reports easier. It is true from a logical standpoint as well, where data elements often do not occur in a record leaving "holes" in the table, or where elements occur multiple times within a record. Some database systems force data into tables when such a model does not reflect reality.

In LBLSTAFF, non-tabular data such as a payroll account number translation, alternate name, and home address are easily accomodated.

7. Variable Length Elements.

The purpose of variable length elements is to avoid storing and processing blank characters so as to conserve storage space, to make output formatting easier, and to avoid having to guess what the longest value ever used will be at the time of the definition of the database. Fixed length elements require knowing *in advance* what the longest value will be, which is impractical, or guessing "long" and wasting storage, or else guessing wrong, too short, and truncating or abbreviating some values. For example, the NAME element in LBLSTAFF contains a record where the value is:

ALEXIS E SCHACH VON WITTENAU

a total of 28 characters. But most names are much shorter. Thus, if the name element were defined as a fixed length element long enough to store the above name, then much storage space would be wasted on all the shorter names. Some systems suppress trailing blanks internally, but then force their restoration when the record is displayed making output formatting inconvenient. For example, to concatenate ELEMENT B to ELEMENT A, as in a bibliographic entry, where ELEMENT A is a fixed length, might require programmer intervention to strip trailing blanks, or worse, user intervention to accomplish a properly formatted output:

Smith, John. How to Build Computers. 1986.

rather than,

Smith , John . How to Build Computers. 1986.

If the name element were fixed length at, say, 28 characters, then blanks would need to be removed to produce a properly formatted bibliographic entry as above. Further, if personal names are stored as separate elements (surname, pre-surname, post-surname), formatting a whole personal name can be more complicated.

Although fixed-length elements are definable in SPIRES, they are rarely used. Though they require slightly less overhead, the inflexibility is usually not worth the small savings. For example, it is tempting to define elements such as LBL EXTENSION or PAN as 4-byte fixed length. But many LBL extensions are off site, requiring 6 or 7 digits. PAN's were defined as fixed length, 4 byte values. This error in foresight became apparent when the Laboratory began using PAN subaccounts.

A less obvious advantage to variable length elements is absence of arcane schemes for abbreviating English words and phrases in data values that usually evolve by necessity in fixed-length storage schemes. Often, only the person who

performed the abbreviation can translate it back into English. Even if the users are familiar with the abbreviations, such data often finds its way into reports printed for public consumption or for managers who are unfamiliar with the terminology, much less the abbreviations.

8. Optionally Occurring Elements.

Most elements are defined in SPIRES as optionally occurring so that storage space is not consumed when there is no element value applicable. As well, it is useful to be able to isolate records on the basis of whether an element occurs or does not occur, or occurs a specific number of times, or occurs but with no value.

For example, an element ALTERNATE NAME is defined in LBLSTAFF so that employees can be searched based on either maiden or married surnames. Yet most employee records have no occurrence of the ALTNAME element and no disk storage is consumed by the element for those records.

9. Multiply Occurring Elements.

Some employees have more than one location, e.g, an office and a laboratory. Multiple occurrences of elements such as building, room, and extension must be able to be individually processed (stored, searched, and displayed), rather than strung together as a single text string. This enables values to be indexed separately for fast searching, and the formatting of each new occurrence of an element to begin automatically on a new line without having to detect where in a string the new occurrence begins. Storing multiple occurrences separately also allows extension numbers to be processed separately, e.g., automatically inserting a hyphen into extension numbers that are six or seven characters, but leaving 4 byte numbers unchanged. Some database systems do not permit multiply occurring elements except as the last element defined.

10. Structures.

To the extent that computer programs coincide with reality, the more useful they are. Often there is inherent logical association among multiply occurring elements. For example, an employee may have two locations, as described above. It is likely that the telephone numbers associated with each are different and that association must be retained. Thus, an element of type STRUCTURE (named LOCATION in LBLSTAFF) logically binds occurrences of elements hierarchically beneath it:

Without Structures:

KEY = 123
NAME = JONES, JOHN
BUILDING = 50
BUILDING = 51
ROOM = 4220
ROOM = 10
EXT = 4444
EXT = 5555
EXT = 6666

With Structures:

KEY = 123
NAME = JONES, JOHN
LOCATION
    BUILDING = 50
    ROOM = 4220
    EXT = 4444
    EXT = 5555
LOCATION
    BUILDING = 51
    ROOM = 10
    EXT = 6666

Structural binding enables all elements in a single structure to be processed together automatically rather than having to perform some procedure to accomplish that task. It is important in displays, especially the printed phonebook to associate (logically bind) extension numbers with their proper location:

Jones, John	9330	50	4220	4444
				5555
		51	10	6666

Rather than (simple line-by-line display):

Jones, John	9330	50	4220	4444
		51	10	5555
				6666

in which case ext 5555 is mistakenly associated with the wrong location.

11. Variety of Data Element Types.

11.1 Stored elements
11.2 Computed elements

11.1 Stored elements
11.1.1 Text
11.1.2 Personal name
11.1.3 Date, time
11.1.4 Integer, real, decimal, packed
11.1.5 Bit, yesno
11.1.6 Dollar (real + rules)
11.1.7 Structure
11.1.8 Executable element values

11.1.1 Text. TEXT is the default element type. Any element value is accepted. Most elements in LBLSTAFF are defined as type text.

11.1.2 Personal Name Processing.

In processing personal names, one has four choices:

- use a single field and enter last name first, which has an undesirable appearance for some applications such as mailing labels;
- use a single field and enter first name first, which yields an undesirable sort order;
- use separate fields for surname, pre-surname, and post-surname data so that they can appear in any order. However, this adds complexity because the portions must be recombined every time the name is processed (displayed).
- use a single field and define processing rules that identify the surname, and allow a variety of output formats.

SPIRES employs the fourth alternative, applying rules (called $NAME, $PNAME) to input, output, index, and search values such that, in the following example, SMITH is always identified as the surname:

John Smith
J. Smith
Smith, John
Dr. John Smith
Dr. John Smith, Jr.
Smith, III, Dr. John
John Smith-Jones

Further, a special $NAME option used in LBLSTAFF allows

Smith III, Dr. John

to be processed properly as well.

Personal name processing in conjunction with other SPIRES facilities enables the following names to appear in the printed directory in their proper sequence in spite of the special characters:

DeHaven
De Jonghe
DelValle
De Marco

O'Keefe, Mike
Oki, Mike

Truncation (wild card) characters allow surname prefix searching without requiring the entire pre-surname portion:

FIND NAME J SMI#

is acceptable. The user does not need to enter:

FIND NAME JOHN SMI#

All of the following will find John Smith's record:

FIND NAME SMITH
FIND NAME J SMITH
FIND NAME JOHN Smith
FIND NAME J Smi#
FIND NAME Smi#
FIND NAME Smith, J

There are other aspects of SPIRES personal name processing, especially regarding indexing and searching, and processing of the non-surname portion, which provide high utility.

11.1.3 Date, time. All common forms of date are recognized upon input and converted to the hexadecimal form of CCYYMMDD. Outprocs (output processing rules) allow for most any form of date to be specified upon conversion to output, including:

01/05/87
JAN. 5, 1987
MON. 01/05/87
MON. JAN 5, 1987
Jan. 5, 1987
Mon Jan 5, 1987
Monday, January 5, 1987
Mon. 05-01-87

1987.01.05
MON. 5 JAN 1987
MON. 1987 JAN. 05

11.1.4 Integer, real, decimal, packed. Stores and displays numerical data efficiently and allows appropriate operations to be performed upon element values.

11.1.5 Bit, yesno. Since any employee or guest might be sent mail or telephone calls, it is essential that both employees and guests be included in the LBLSTAFF subfile. Further, those persons might be sent mail or telephone calls even after termination, and so their extension and mailstop must be retained so that another staff member in the group or department can transact Laboratory business as needed. When an employee is terminated, a termination date is entered in their record, rather than removing the record from the database.

Thus, not all persons represented in the LBLSTAFF database should appear in the published telephone directory. A YES/NO flag is used in LBLSTAFF to distinguish whether a record should appear in the published LBL directory. This control is provided by the PUBLISH element in each record which is required to have a value of either YES or NO. Non-terminated employees are assigned a YES value. When the terminate, the value is set to NO. Guest records are assigned PUBLISH = NO, except where their sponsoring department or division requests that they appear in the directory.

11.1.6 Dollar. Special characters (dollar signs) are stripped on input and stored as type real. Special characters restored upon output.

11.1.7 Structure. Structures are discussed above (IV.10)

11.1.8 Executable element values. Allows elements to be interpreted as commands.

11.2 Elements Computed rather than stored:

11.2.1 Virtual. Creates a value either by operating on values of other elements or system variables, or by generating a value in a programmer-defined manner.

11.2.2 Redefined Virtual. A type of virtual element that redefines an element that exists in the database. E.g., in TRAINING, the virtual element PAN redefines the employee ID to be the PAN by accessing OCTOPUS.

11.2.3 Dynamic. Allows a user to define an element derived either from other elements or from variables, which exists for the duration of an interactive session. For example,

DEFINE ELEM ANNUAL SALARY AS @SALARY * 12

Dynamic elements are used occasionally in LBLSTAFF for ad hoc report making.

11.2.4 Phantom Structures. A type of virtual element that enables a record or part of a record in one subfile to appear as if it were part of a record in another based on some element-in-common in an output format. For example, in the TRAINING subfile, the data describing courses appears in student record as if they were part of the student record.

12. Case Handled Intelligently.

Data can be entered and stored in mixed or uniform case. Input rules (INPROCS) can be specified to force all input values to a specified case, e.g., first letter upper - the rest lower, all upper, etc. Regardless of the case in which goal record data *is stored*, index values and search values are forced to upper case, by default, so that the user need not know the case for searching (although this feature is overrideable):

FINd NAME l smith will find: L Smith, l smith, L SMITH, etc.

13. Word Indexes.

By coding $WORD in the file definition PASSPROCs and SEARCHPROCs, the element value string will be stored as a single text string in the goal record, but will be passed to its index as separate substrings delimited by blanks or other designated characters (i.e., words) rather than as a single string.

For example, the ELECTRONIC MAIL element in a record might appear:

EM = KONRAD AT UCBCMSA

It is desirable to have the following searches all successfully retrieve the record:

FIND EM = KONRAD
FIND EM = UCBCMSA
FIND EM = KONRAD AT UCBCMSA
FIND EM = KONRAD UCBCMSA
FIND EM = UCBCMSA KONRAD

PASSPROCs and SEARCHPROCs can convert 'AT' to null and '@' to space for indexing and searching.

14. Protocols Language Tools.

SPIRES protocols language provides general language tools such as if-then-else branching, all SPIRES query commands, system variables, user-defined variables, report making commands, the ability to read from and write to CMS files, and the capability to prompt the user, then store the response in a variable which can be tested or otherwise processed. Of course, the same functions can be accomplished in common programming languages such as COBOL or FORTRAN, but they lack built-in "knowledge" of SPIRES data and structure, and would require much more programming effort.

The SERVICE subfile uses a SPIRES protocol named FASTXEQ to prompt telephone and mailroom attendants for search values so that they do not need to know any SPIRES query language.

15. System Procedures, Functions, and System Variables.

As mentioned, a database system is a collection of preprogrammed software tools. SPIRES provides many such tools in the forms of system procs, system variables, and system functions.

System procedures (PROCS) are actions that can operate on values entered into SPIRES, output by SPIRES, passed from a goal record to an index, or values specified by a user in a query. They enable customized processing to occur using pre-programmed tools which save programmer time and owner-defined tools which add flexibility:

INPROCS	Actions executed when the record is input (or updated)
OUTPROCS	Actions executed when record is displayed or printed
PASSPROCS	Actions executed on an element value before it is passed to an index
SEARCHPROCS	Actions performed upon a search value before the index is searched
USERPROCS	User-defined procedures for input, output, passing, or searching.

A USERPROC can be written to process a value based on either another element value or the current value of some system or user variable, or other source. This is important for sequencing names properly in LBLSTAFF for the published phonebook. Some names contain blanks, apostrophes and other special characters. Records to be published in the directory are sequenced, not according to the value of the NAME element, but according to a virtual element called SEQN ("sequence name") which processes the NAME value by calling a USERPROC:

```
Element = Name;
Inproc = $NAME.SPECIAL(comma);
Outproc = $NAME.SPECIAL(comma);

.
.
.
```

Virtual elements:

```
Element = SEQN;
Outproc = $CALL(CANON)/ $CALL(NICKN)
REDEFINES = NAME


USERPROCS;
USERPROC = CANON;
     Let SQN = $cap($value)
     Let SQN = $change(#sqn, "","")          removes apostrophes
     Let SQN = $change(#sqn,-,9)             changes hyphen to 9 for sequencing
     Let SQN = $change(#sqn,JR.,null)        removes JR.
     Let SQN = $change(#sqn,SR.,null)        removes SR.
     Let SQN = $change(#sqn,III,null)        removes III
     Let SQN = $change(#sqn,II,null)         removes II
     etc.
     Set VALUE = #sqn
```

```
USERPROC = NICKN
    let m = $match($value,?'((')          scans string for double paren which
                                          indicates a nickname is included in
                                          name string
    If #m = 0 : return                    If 0, no nickname in string
    let surname = $break($value,',')
    let nickname = $break($value,'((')
    let nickname = $strip($value,#nickname)
    let nickname = $change(#nickname,'))'null)
    let nickname = $change(#nickname,'((',null)
    set value = #surname',',#nickname
```

Therefore, the names indicated in Section IV.11.11 will appear in the printed phonebook in the proper sequence despite the special characters.

Examples of INPROCS, OUTPROCS, PASSPROCS, and SEARCHPROCS:

$break	Splits value into multiple occurrences at delimiters
$call	Calls a programmer-defined procedure
$date $date.out	Verifies date value, converts to hexadecimal converts hex value to specified output format (Section IV.11.13)
$Default	Generate default values
$dollar $dollar.out	Converts money text string to floating point converts floating point to money text
$getelem	Fetch a specified occurrence of an element value from a specified record
$GETUVAL $GETCVAL	Navigates the record structure to fetch specified value (internal or external version of value)
$length $max.len $min.len	Verify length of element value
$lookup $subf.lookup	Fetch a data element value from a record in another subfile or record-type.
$max.occ $min.occ	Verify number of occurrences of data element value
$msg	Specify text of error message in event error occurs.
$name $pnames	Convert personal names for processing
$range	Verify that value is in a specified range
$search.subf	Perform indirect searching (virtual index)
$search.trunc	Allow truncated searching on an index
$time	process time value
$yesno $yesno.out	Flag processing

Especially important are the text-oriented actions:

$break	Splits value into multiple occurrences at delimiters
$cap	Capitalizes words or entire values
$lower★	Changes value to lower case
$case	Provides several options for specifying case
$change	Changes a character string to another character string or null e.g., $change(hello,goodbye) changes "hello" to "goodbye".
$exclude	Discards values in a list, usually for indexing and searching, e.g., "at, the", etc.
$include	Process only those values in a list, e.g., index only months of the year from an element value.
$insert	Insert a text string before, within, or after a value.
$insetl $insetr $insetc	Left (right, or center) justify a value and fill to make a value of size n E.g., $insetl(hello,+,10) returns "hello+++++"
$LSTR $Rstr	Returns a substring of the input value truncated to its leftmost (rightmost) portion for a specified integer number of characters. E.g., $lstr(find,3) returns "fin"
$LSUB, ★ $RSUB★	Returns a substring of the input value truncated to the left (right) of a specified substring, e.g., $lsub(select,e) returns "s"
$match	Compares input value with a predetermined list of string patterns and returns the corresponding integer, e.g., $match(Z,X,Y,Z) returns "3".
$pmatch	Acts like $match but for matching against exact stems
$Substr $Xstr	Extract string from within a value.
$size★	Returns the length of the value in bytes
$squeeze	Removes leading, trailing and multiple blanks.
$Strip	Returns substring of the input value beginning with first character found in the value and not in a second specified string, e.g., $strip(book,oeb) returns "k"
$SPAN	Returns substring of the input value broken to the left of any character found in the value and *not* in the second specified string, e.g., $span(book,oeb) returns "boo".
$verify	Verify that value is of a specific type (real, integer, text, hex, date, etc.)
$word	Separate value into individual occurrences at blanks or other specified delimiter

★ indicates this function also available in AT&T CSM

System variables provide access to useful information about SPIRES both interactively and in SPIRES protocols. There are many, the most often used including:

$Account	Returns the user's logon ID.
$Ask	Stores the most recent response to the ASK prompt
$Date, $time	Current date and time
$Filename	Name of currently selected or attached SPIRES file
$Format	Name of currently set format.
$Select	Name of currently selected subfile
$subfile size	Number of records in the subfile
$key	Key of goal record most recently processed
$Lastcmd	Stores the last command executed
$result	Number of records in the most recent search result
$yesno	Indicates whether the previous command failed.

There are many others for specific functions, such as output formatting:

$CCOL	Stores the column that will be written to next
$Crow	Stores the row that will be written to next
$cval	Converted value of most recently retrieved element
$Wdsw	The line most recently written (e.g., the 15th line)
$wdsr	The line most recently read
$idata	Name of input file
$odata	Name of output file
$pageno	Last page number written

etc.

16. Ease of Use for Adding, Updating, Removing Records.

As with many systems, element-by-element prompting is available for adding and updating records. But it is not mandatory to step through an entire record in that manner. SPIRES also allows a user to transfer a whole record into a user file where it can be edited with a full screen editor.

Further, the full screen editor enables occurrences to be repositioned relative to each other with out reentering the data (as in Datatrieve). Suppose a record appeared:

```
KEY = 1
NAME = TOM SMITH
LOCATION
    BLDG = 50
    ROOM = 222
    EXT = 5555
    EXT = 4444
    EXT = 6666
```

and suppose that it was desired that the 6666 extension become the primary extension. The line can simply be moved in the editor and the entire record returned to the database.

It is helpful in editing LBLSTAFF location structures with multiple extensions to be able to combine or separate a structure on a full screen, i.e., to have access to the whole record simultaneously, and then simply return the edited record to the database with the UPDate command.

17. English-Like Syntax.

SPIRES syntax is intuitive:

show private subfiles (show what private databases can be selected)
SELect <subfile> e.g., sel lblstaff
sho formats (lists custom formats available)
Show elements
Sho subfile size (shows number of records in the subfile)
sho subfile information
Sho select (shows name of subfile selected)
Sho result (shows size of the search result)
sho indexes
sho element information
browse <index name> <value> e.g., browse name smith shows the "smiths" in the name index
Find name dave shirley and bldg 50
set format snarf
sequence bldg room name (resequences records a search result or a stack)
Type
for subfile where location occurs > 1
Update
clear format
for updates where publish = NO
display all

18. Security.

Various security features are provided in SPIRES.

18.1 Deferred Queue. It is essential in database updating, as in text editing, to have some means to undo a change and to return to *any* previous version of a record. If, for example, an incorrect value has been entered in one or more records, or records have erroneously been deleted from the database, it is important to be able to restore the prior state. As well, it is useful to be able to obtain a view of "all records updated today" or "all records added today". SPIRES provides this capability by storing all versions of a record made "today" in the Deferred Queue which allows access to each version as well as access to the unchanged and unremoved version of the record. The current version is always presented by default.

18.2 Concurrent Updating Control. It is essential in a multi-user database that no user's modification of a record interfere with or "wipe out" a simultaneously updated version of the same record by a different user. This is important in LBLSTAFF where several accounts have permission to update records. SPIRES handles concurrent updating appropriately with the use of SECURE-SWITCH 10 being SET in the file definition:

- A user TRAnsfers a record to a physical (CMS) file which can be edited. The record is not removed from the database during this time. It is still displayable by other users. The TRAnsfer command transfers not the record, but *control of* the record to the user who is updating it and a copy to a CMS file where it can be edited.
- If another user attempts to update the record by issuing a TRAnsfer (or MERGE under prompting), a message is sent to the second user indicating that someone is presently updating the record and prevents transferring control of the record to his account.
- When the user updating the record has finished editing, he issues the UPDate command which sends the new version of the record to the database as the current version, and enables any other individual who has permission to TRAnsfer the record to perform updates of his own.

18.3 Recovery After a Day. If the erroneous modifications by an authorized user are not discovered the same day, i.e., while they're still in the deferred queue, the CMS RESTORE facility can provide earlier versions of CMS files. Thus, the database can be restored to appear as it did on almost any previous day. At UCBCMSA, the tape backup facility runs continuously, backing up changed files nearly every day.

18.4 Recovery After a System Crash. If the computing system crashes during an add, update, or overnight processing, it is important that the database system be able to recover gracefully. SPIRES recovers from such incidents, usually transparently, by use of internal checkpoints. Occasionally the file owner is required to issue a PROCESS command to complete the interrupted processing. SPIRES users do not lose data in the event of a system crash unless they are in an editor when the system does down. In that event, the data has not yet been entered into SPIRES and thus SPIRES cannot save it. CMS might retain a portion of the edited file.

18.5 Recovery During Disk Crash. Occasionally disks may become unusable. When this occurs to LBLSTAFF, it is simple to modify an entry in FILELOC, a SPIRES database that tells SPIRES which CMS files are to be considered as the files that comprise a particular database, to point to backup copy of the database. This occurred in December, 1986, and the database was available in less than 10 minutes after the crash. Later that day, the data on the bad disk was restored by UCMCMSA staff. It was not necessary to use the CMS RESTORE facility which provides system backup and restoration, but that too was available had it been needed. No SPIRES data has ever been lost at UCBCMSA.

18.6 Security at the Element Level. Users have persmission to see or not to see, update or not update, and to search or not search each data element in a SPIRES subfile individually by element, individually by user, if desired. SPIRES employs the mechanism of a PRIV-TAG associated with each element and linked to an accounts list to identify which users can process which elements and what processing is allowed. This feature is used extensively in LBLSTAFF.

18.7 Security at the Subfile Level. It is important to be able to control access to the each subfile, which SPIRES accomplishes by allowing specified accounts for each subfile access. For example, only the Mailroom, Telephone Services, and Shipping and Receiving can SELect SERVICE to get rapid employee location displays that include termination dates.

18.8 Data Validation. It is essential to be able to control element values. For example in LBLSTAFF, mailstop values must occur in a lookup table record in the MAILSTOP subfile before they can be used in an employee record. This reduces the chance of spurious mailstop values from being entered into employee records. Many other types of data validation are available, including the capability for the file owner to write their own procedure that is compiled into the file definition, a USERPROC, that is invoked each time the element is processed in a record.

18.9 Subfile Logging. For SPIRES subfiles for which "Command Logging is in effect", every command entered by a user, and the date and time are entered in the log. This is especially helpful in assisting new users.

19. Report Generators.

SPIRES provides three means of output formatting:

First, the default SPIRES output format, which is used for most updating, is an element-by-element listing in the form:

(Element name) (optional equal sign) (element value) (semicolon)

E.g., ID = 123;
Name = John Jones;
Location;
Bldg = 50;
Room = 130;
EXT = 4444;

The second form of output formatting is invoked interactively by setting format $REPORT, which produces a table:

SELect LBLSTAFF
FIN PAN 9191
SET FORmat $REPORT ID(2,7) NAME(10,30) pan(45,4) EXT(50,10)
TYPE (or IN ACT CLR DIS ALL to put the result in a file)

ID	Name	pan	EXT
E111111	Fink, Robert	9191	4444
E222222	Konrad, Allan	9191	5555
			7091
E333333	Rogers, Sig	9191	6666

etc.

Format $REPORT is used to download formatted LBLSTAFF records to an LBL Vax for distribution to other systems.

Finally, customized formats are programmable to allow specific processing and placement of data, general language features such as looping, if-then-else, etc., and creation, modification, filtering, and insertion of any other information into a report whether it is derived form data stored in a database or not. For example, a customized SPIRES format inserts *troff* command language into formatted LBLSTAFF records to produce the alphabetical portion of the printed phonebook.

A common use for customized output formatting in LBLSTAFF is production of adhesive mailing labels. For example, to make adhesive mailing labels for OCR staff:

SELect LBLSTAFF
Fin pan 9191 and publish = y
set format label
in act clr type
LABEL active file a (this sends the output file to the LBL label printer via BITNET)

Coupling to graphics and statistics packages. SPIRES system staff at Stanford have chosen not to duplicate graphics and statistical capability provided by other sophisticated packages, but rather to make interfacing with those packages easy. SPIRES custom formats and protocols provide the desirable user friendliness.

20. Large Capacity.

SPIRES databases can span several physical disks in VM/CMS. Currently the largest SPIRES database at LBL is the Mechanical Engineering Drawings database of 200,000 records. Each element has about a half dozen data elements such as drawing ID, category, type of drawing, and title, date entered into database. The database is about 20 MB.

V. Remote SPIRES.

Remote SPIRES is a facility which runs under CMS in a disconnected virtual machine in conjunction with SPIRES to provide fast, simple database search and retrieval either locally or from any authorized BITNET user.

Search and report-making commands are sent via BITNET. In the BITNET environment there are three types of entities that a user can send or receive: files, mail (a file with header information), and messages (which interactively appear on the screen of the addressee). From a VM/CMS environment, Remote SPIRES can accept commands via messages using the TELL command and via mail using the NOTE command, using the format:

TELL user AT node query (IN subfile

or

NOTE user AT node

The interactive message form is preferable,

For example, using interactive messages,

TELL QSPIRES AT SLACVM FIND author A. Jones (in HEP

would find all preprints where A. Jones occurred as an author in the High Energy Preprints file maintained at SLAC.

LBL databases will be similarly searchable:

TELL LBLREMOT AT UCBCMSA FIND NAME R. SMITH (IN PHONEBOOK

will find R. Smith's record in LBLSTAFF.

TELL LBLREMOT AT UCBCMSA DISplay 80A (in BLDG

will who the building manager is for building 80A.

On workstations, the prefix TELL LBLREMOT AT UCBCMSA can be set with a function key so that one keystroke enters the whole string.

On the LBL Central Computing Facility Cluster, the user must be on CSA2 to send BITNET messages. The command BSEND is used rather then TELL as in CMS. BITNET messages cannot be sent from LBL unix machines at present. Separate instructions will be made available for LBL users who do not have access to CSA2 and must therefore use mail.

VI. Costs.

The annual SPIRES software maintenance fee is $7,000 which covers the right to use the newest version, media (9-track tape), full set of documentation (on tape), and system consulting support from the Consortium office. This fee has been paid by OCR for several years.

The monthly costs for LBLSTAFF and other associated SPIRES accounts, including storage costs, online charges, and batch charges (but not local LBL Computing Facility charges) for September and October 1986 were:

Account	Usage	September	October
TPHHH	Stores LBLSTAFF, used for updating	$ 175.93	$ 149.98
INFORTPH	Telephone Attendants 7 am - 5 pm	149.63	145.48
MAILR	Mailroom, 7 am - 5 pm & weekends	180.76	156.70
SSRHP	Shipping & receiving	73.71	77.28
SSDVN	D. Nielsen	21.94	32.28
ADHDVN	D. Nielsen	4.21	16.72
SPIRES	System files	343.28*	288.64*
SFSYS	Fileserver	359.92*	309.84*
LBLOVNT	Overnight Processor (batch)	40.03*	39.20*
SEALY82	Training & Emergency Preparedness	47.53	119.77
KOEHN	RPM database	31.39	7.65
KONRAD	Consulting, system maint, applications	87.87*	79.11*

* Covers service for applications other than LBLSTAFF

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720