AT&T Technical Journal
Vol. 64, No. 1, January 1985
Printed in U.S.A.

System 75:

System Management

By H. K. WOODLAND,* G. A. REISNER,* and A. S. MELAMED†

(Manuscript received July 11, 1984)

System management is the task of installing, administering, and maintaining a communications system. Customer and technician access to software-based administration and maintenance capabilities in the System 75 office communication system occurs through an on-line video display terminal called the System Access Terminal (SAT). With the SAT, the user may install, test, rearrange, and change equipment and services. The SAT hides the internal complexity of the system while presenting all the capabilities in as simple a manner as possible. A layered software architecture is used to perform data view mapping from the user view to the internal data representation.

I. INTRODUCTION

An integral part of System 75's enriched feature set is its capability of allowing the user to install, test, rearrange, and change equipment and services, and select a large number of per-user and system feature options. The user accomplishes this by entering and modifying data in the system's distributed, *translation* database. Also available are system-generated measurement reports, translation data backup on cartridge tape, and bulk data transfer (translation data and program updates) to AT&T Information Systems Operations Support Centers.

*AT&T Information Systems Laboratories, an entity of AT&T Information Systems, Inc. †AT&T Bell Laboratories.

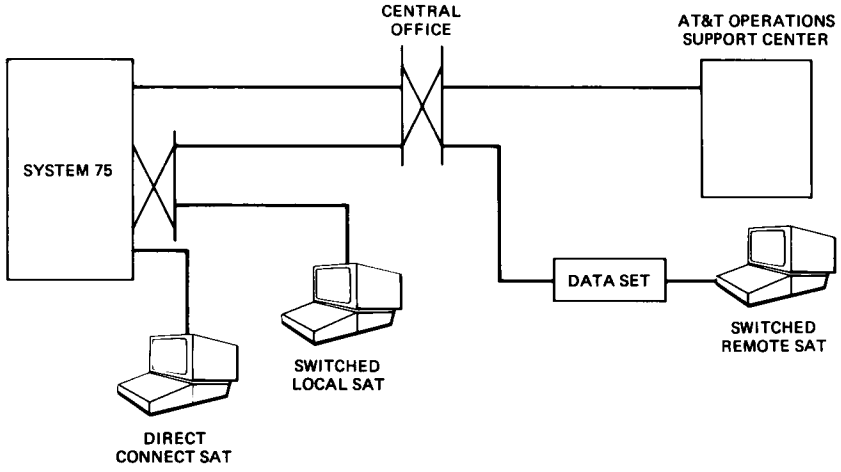

Fig. 1—System management access.

One objective of the system management task of System 75 is to allow customer participation by providing an interface that can be used by customer personnel as well as AT&T Information Systems technicians. This interface hides the internal complexity of the system while presenting all the capabilities in as simple a manner as possible.

This article discusses the key concepts of this user interface and the layered architecture that makes reliable translation database management possible.

II. TERMINAL USER INTERFACE

2.1 System access

System 75 operations and training are simplified by providing a single system-management user interface. Customer and local technician access occurs through an on-line video display terminal called the System Access Terminal (SAT).* The SAT (the 513 or 515 BCT) may be:

1. Directly connected to an RS-232C EIA port,
2. Switched through a Digital Communications Protocol (DCP) dial-up port, or
3. Switched through a modem dial-up port for off-premises access (Fig. 1).[1]

The local SATs operate at 9600 baud while the remote SAT—connected via a Direct Distance Dialing (DDD) line—operates at 1200 baud. In addition to the SAT interface, a 1200-baud, X.25, host-to-

*Acronyms and abbreviations used in the text are defined at the back of the *Journal.*

host interface is provided to accommodate the remote AT&T Information Systems Operations Support Center.

2.2 Terminal user access

Access security, an important aspect of any shared access system, is maintained through the use of login names and passwords. Repeated invalid passwords will cause the system to drop the line and increment a security violation count.

Multiuser access is also important. System 75 allows two SAT users and one remote operations host to be logged in at the same time. User contention is managed on a per-command basis and only one user at a time may execute a command that changes the database. This eliminates the confusion that exists when two users interact with the same data. These design requirements greatly simplified the architecture and implementation.

2.3 Terminal interface screen layout

A simple, consistent, easy-to-understand screen layout is an essential element of a good user interface. The System 75 terminal interface screen is divided into four regions (Fig. 2):

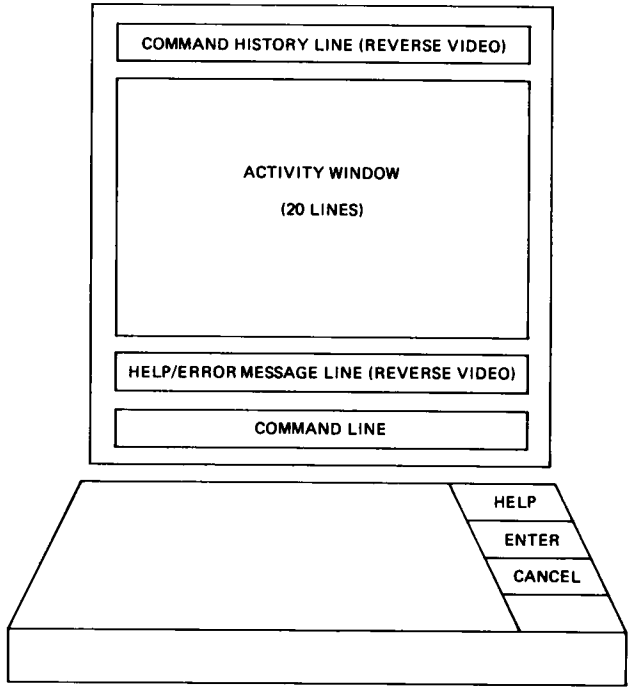

Fig. 2—Terminal interface screen regions.

1. A command path or *command history line*
2. A 20-line command output or *activity window*
3. A *help/error message line*
4. A *command line.*

Reverse video is used in alternate regions to partition the screen layout without wasting valuable screen lines.

2.4 Command entry

Command organization is another important aspect of a good user interface. Upon logging in, the System 75 user is placed in a single-level *command entry state.* All commands are directly accessible from this state. There is no tree structure to traverse or mode keys to depress to move among administration and maintenance commands.

Commands are logically divided into ten "administrable" categories. Each user login name is allowed or denied permission to execute commands on a per-command category basis. Confusion is eliminated because a user only sees those commands that he has permission to execute. As the need and desire for customer participation increases, the customer may be given permission to execute more commands.

The commands utilize English-like phrases. The format is *action, object, qualifier.* For example, the command to add a station set with the extension number 3261 is:

```
add station 3261,
```

where `add` is the *action* to be performed, `station` is the *object* being acted on, and `3261` is the *qualifier.* Complete or partial commands may be entered providing the user enters enough characters of a parameter to distinguish it from other legal parameters. For the example above, the command could be entered as:

```
a s 3261.
```

The system prompts the user with on-line messages and, at any time, the user may request assistance by depressing the *help* key. The response from the system depends on the parameter type to be entered: if an *action* or *object,* then a list of valid actions or objects is displayed; if a *qualifier,* a message is displayed describing the format and type of data to be entered. For example, if the command was

```
add station 20,
```

the system would respond with the message

```
20 is an invalid entry—Please Press Help.
```

If the *help* key were depressed, the system would respond with the message

```
Enter a number between 100-9000,
```

where `100-9000` are valid extension numbers. This sort of prompting greatly reduces training time and paper documentation. Other actions include `display`, `change`, and `remove`.

2.5 Command output

Three basic types of output appear in the *activity window*:

1. A status report,
2. A data entry form, and
3. A restricted form.

A *status report* is produced when a `display` command is entered, a data entry form is produced when an `add` or `change` command is entered, and a restricted form is produced when a `remove` command is entered. In the *data entry form state*, the user may modify the data and then store away the changes by depressing the *enter* key. In the *restricted form state*, the user is prohibited from modifying the data on the screen. The data are displayed only to allow the user to verify that the correct data are being removed. The user may depress the cancel key to abort any operation.

2.5.1 Data entry form state

The *data entry form state* allows formatted, forms-based data entry. Forms are associated with a specific task. For example, the station form is used to display the precise data required to add a station set (Fig. 3). Fields are added or removed from the form dynamically, as required. When the data entry form state is invoked, a form appears in the activity window with data fields defaulted to the most commonly

```
add station 3261                                          Page 1 of 1

                                    STATION

 Extension:  2000            Station Type: 7303S                Port: _____
      Name: _____________      Coverage Path: __                   COS: 1_
                               Security Code: ___                  COR: 1_
FEATURE OPTIONS
            LWC Reception? y          Coverage Msg Retrieval Permission? y
          LWC Activation? y                          Data Restriction? n
     Redirect Notification? y                 Idle Appearance Preference? n


ABBREVIATED DIALING
     List1: _______              List2: _______              List3: _______

BUTTON ASSIGNMENTS
  1: call-appr                          6: _________
  2: call-appr                          7: _________
  3: call-appr                          8: _________
  4: _________                          9: _________
  5: _________                         10: _________
```

Fig. 3—System management station form.

used values. The command entered and the number of form pages appear on the *command history line.*

Standard cursor control keys (arrow, next/previous field, clear field, next/previous page, and refresh screen) allow the user to move about the form in a prescribed way. In the station form, the *next field* key moves the cursor left to right in the upper part of the form, and down the columns in the bottom (button assignment) part of the form. This guides the user to complete the form in an orderly manner. However, the *arrow* keys provide the flexibility to move the cursor in any desired direction.

2.5.2 Data validation

In some systems, data are validated only after all data have been entered. Where possible, System 75 performs validation immediately as data are entered or changed (see Section 3.4). This allows immediate response to many user errors, thereby decreasing the overall user time required to enter and validate a given form.

2.6 Special commands

Two special commands were added to assist the user in adding objects to the system. One handles the case when the user wants to add an object and does not know the next available qualifier. For this case, the `add object next` command is provided. For example, if `add station next` is entered, the system searches the database, selects the next available extension number, and invokes the data entry form state with the selected extension number.

The *duplicate station* command speeds the addition of a station by copying data from an existing station. For example, if `duplicate station 3261` is entered, the system invokes the data entry form state and displays a station form with data identical to those of station 3261, except that the extension, port, and name fields are blank. These differentiating fields may then be entered to create the new station.

In summary, the SAT provides a simple, easy-to-understand user interface that may be used by the AT&T Information Systems technician and trained customer personnel.

III. SYSTEM MANAGEMENT ARCHITECTURE AND DESIGN

The system management software provides four functions, all of which are available through the SAT:

1. Measurement collection and reporting
2. Maintenance testing and reporting
3. Translation data backup on tape
4. Translation database management.

The measurement collection and reporting capability provides

hourly traffic data on engineered resources, e.g., trunk groups, which are made available through formatted reports. Maintenance reporting and testing capabilities include the demand testing of circuit packs and terminal equipment and the display of system error and alarm logs. Translation data backup on tape provides system translation data backup on a secondary storage medium. Underlying all system management functions is translation database management. The remainder of this article concentrates on this topic, and the software designed to support it.

Translation database management software provides four important functions: data view mapping, database validation, form transactions and concurrency control. Data view mapping allows a user to display and change all translation data related to a single task (e.g., station installation), while hiding the complexities of the internal data organization. Database validation ensures that data entered into the system are individually correct and consistent with respect to other data. For example, validation ensures that extensions assigned to stations are consistent with the dialing plan. Transactions ensure that all the translation data entered on a system form are either valid and accepted or inconsistent and rejected. Concurrency control allows there to be multiple SAT users and, at another level, ensures that switch services software will not use critical data that are being changed. Before describing how these functions are accomplished, we provide a brief overview of the system management software structure. The software consists of three layers:

1. User interface and control
2. Command execution and validation
3. Data access and storage.

The *user interface and control* layer provides users with access to the system, through either the SAT or the X.25 remote link. The *command execution and validation* layer and the *data access and storage* layer provide a single, record-based interface to the user interface and control layer, independently of the access method. Figure 4 shows a block diagram of the essential software layers and components.

The *command execution and validation* layer consists of four software modules, each of which supports the different logical functions described above: (1) measurement collection and storage, (2) administration database update/validation, (3) maintenance command execution, and (4) translation backup on tape.

The *data access and storage* layer consists of the administration database access module and all processes which store translation data.

3.1 Data view mapping

Translation data in System 75 are distributed. They are contained

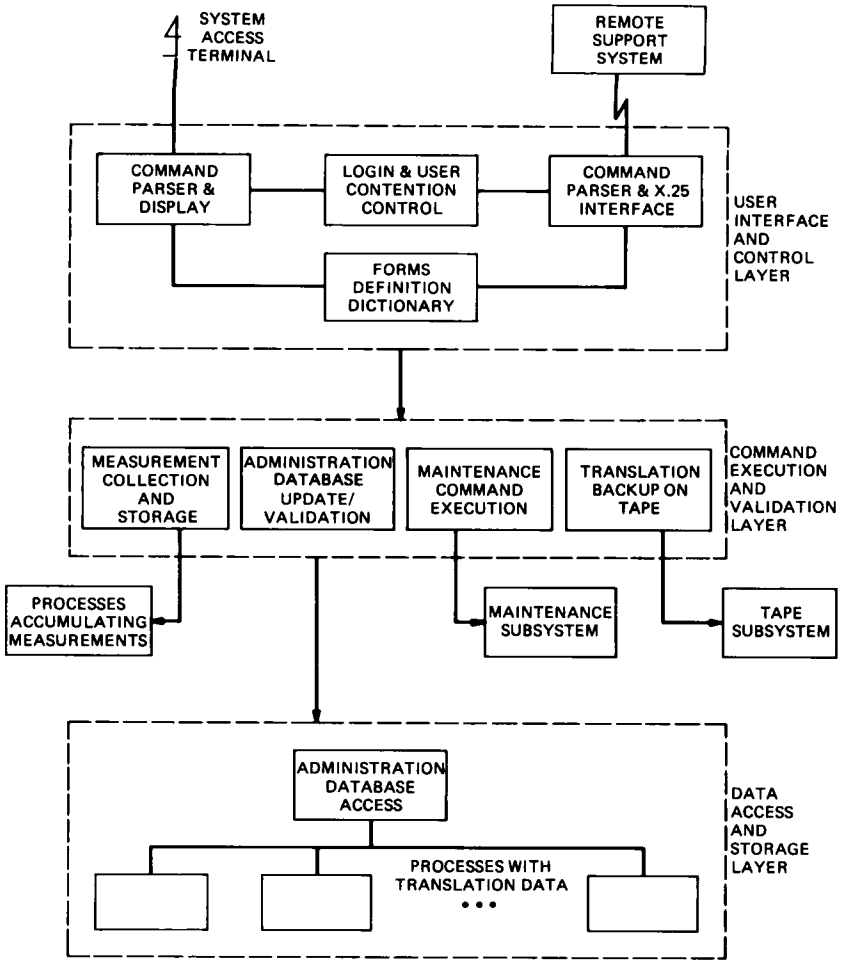

Fig. 4—System management layered software.

in the local data space of the processes that access them in order to meet real-time requirements for voice and data switching. This results in the fragmentation of logically related data. System management software maps the data entered on forms (the logical view of translation data) into the internal view of data, as required by the switch services architecture.[2]

3.1.1 Low-level view of data

The administration database access module distributes and retrieves translation data using a record-based interface. This is accomplished by a set of C language library functions that are supported by each

process storing translation data. This interface is akin to the data manipulation language of database management systems. At an abstract level, however, this interface is designed to be less powerful than a true database management system (e.g., relational manipulations on internal tables are not supported). The reason for this choice of design is to reduce complexity and generality of access methods, thereby increasing the speed of data retrieval.

A simplified view of low-level data view mapping is shown in Fig. 5. It consists of two parts: mapping external identifiers to internal identifiers and distributing data values to the processes storing translation data. This concept is illustrated by looking at some of the data associated with a station set (see Fig. 6). The external identifier for a station is its extension, i.e., the number dialed to call the station. Some of the data associated with this station are the type (digital), name of the person assigned to the station (John Doe), and various buttons.

The internal identifier for the station is an ordered pair (TYPE, INDEX), where TYPE=STATION identifies the station table. This internal identifier is generated by finding the first empty slot in the station table. The internal identifier is then used as a key to identify other data for the station in the button table and the miscellaneous data table. The mapping from extension to internal identifier is contained in the extension table (this extension-to-internal identifier table is used whenever a station extension is dialed).

In summary, the mapping between internal and external identifiers is accomplished completely within the data access and storage layer.

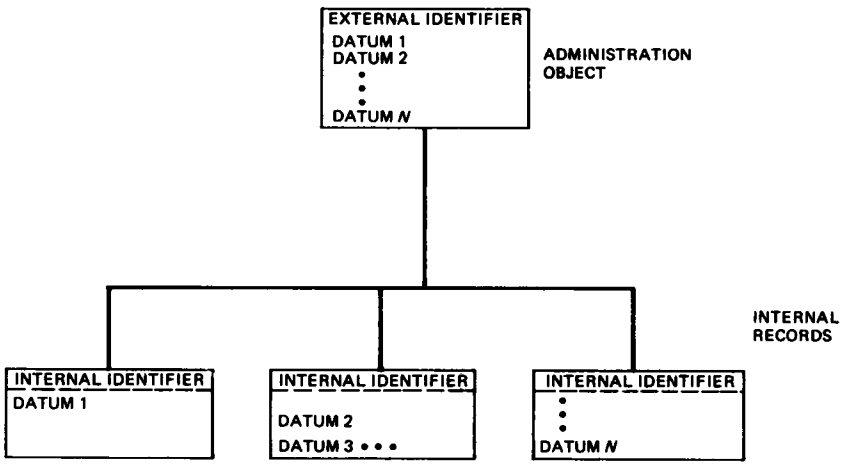

Fig. 5—Simplified low-level data decomposition.

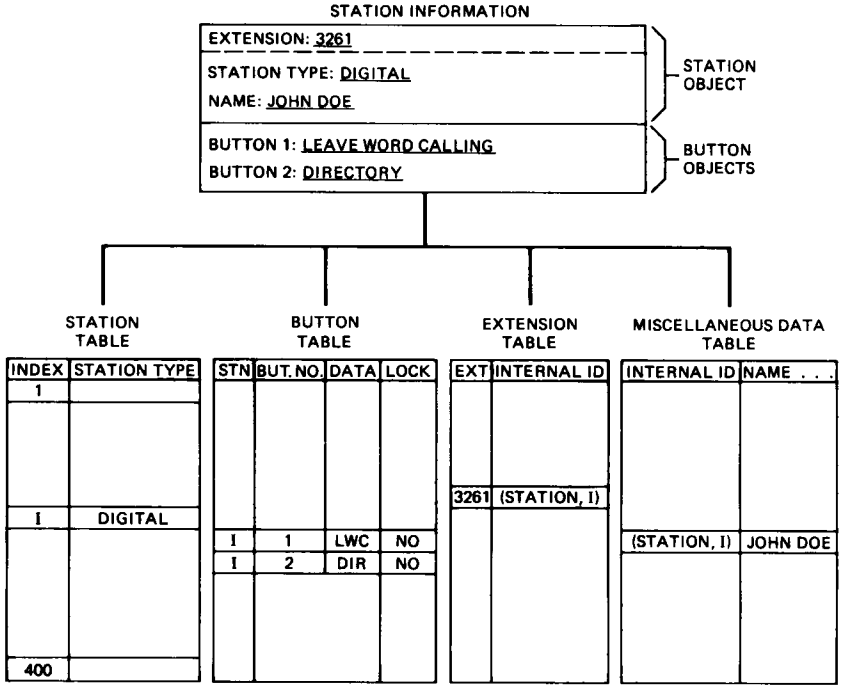

Fig. 6—Data decomposition for a station.

The upper layers of system management software use external identifiers exclusively.

3.2 Concurrency control

As in any database system with multiple users, System 75 must provide a means of managing concurrent updating and a means of managing simultaneous reading and writing of translation data. The first concern, concurrent updating, is handled at a very high level in the user interface and control layer. The login and user contention control module tracks each user logged into the system and each command executed. In this way, it assures that only one user at a time is trying to update translation data. It performs this control by allowing only one **add**, **change**, or **remove** command to be executed at a time. Thus, the problem of multiple concurrent database updates is eliminated.

3.2.1 Locking of internal records

The second problem, managing simultaneous reading and writing of data, is handled in the data access and storage layer. To solve this problem System 75 implements read/write locks on internal records.

In any system using locks, there is a trade-off between speed and the overhead of the locking mechanism. Because System 75 is a real-time switching system, and because data retrieval involves the very same processes that are accomplishing switch services, locks are used with a small, selected subset of internal records, thereby eliminating unnecessary performance penalties.

The mechanism for locking is illustrated in Fig. 6, the station data decomposition example. A read/write lock is associated with each button in the database. This lock serves two purposes. First, it prevents the switch services software from using button data while the button is being changed. This assures that the status data maintained for each button/lamp is consistent with the translation state. The consequences of inconsistencies could manifest themselves, for example, in a lamp that is stuck in the lit state permanently (for a button that no longer has a function associated with it), or until the station is reset by unplugging and reinstalling or until fixed by an internal system audit. The second purpose allows changing an individual button on an active station as long as that particular button is not in use. This granularity of locking provides a great deal of user flexibility in changing data while also guaranteeing a high degree of data protection. Locking also supports a database transaction mechanism described in the next section.

3.3 Database transactions

Administration of the application database takes place during normal system operation. System 75 uses a database transaction mechanism to guarantee that the user command associated with a form is either processed to completion or not at all. In database terminology, transaction management maintains the database consistency, permanency, and atomicity.[4] The capability to abort transactions exists. When one of the requests of a transaction cannot be performed, an automatic rollback of the partially completed transaction occurs. The same procedure is followed in case of system recovery, provided a database transaction was in progress.

The system management software places *exclusive* locks and the switch services software places *shared* locks on data. The first request in a transaction encountering a locked resource causes the system to back out of the transaction, thus avoiding deadlock. A two-phase locking protocol is used. All the internal records required by a transaction are locked before any locks are released. The locks are released only after the entire transaction has been processed.

Transaction requests are generated by the command execution and validation layer, and the actual transaction mechanism is implemented

in the administration database access module. A typical transaction consists of multiple requests. For example:

```
BEGIN_TRANSACTION
UPDATE STATION (3261)
UPDATE STATION_BUTTON (3261, BUTTON 1)
UPDATE STATION_BUTTON (3261, BUTTON 2)
END_TRANSACTION
```

A single request from the command execution and validation layer may map into multiple operations on internal records. When a single request is received, the administration database access module dynamically allocates an instance of an internal record execution table, which depends on both the *object* of the request (e.g., station) and the *action* of the request (e.g., update). Figure 7 illustrates the transaction execution table for the transaction above. The execution table consists of a dynamic and a static part. The static part describes the sequence of

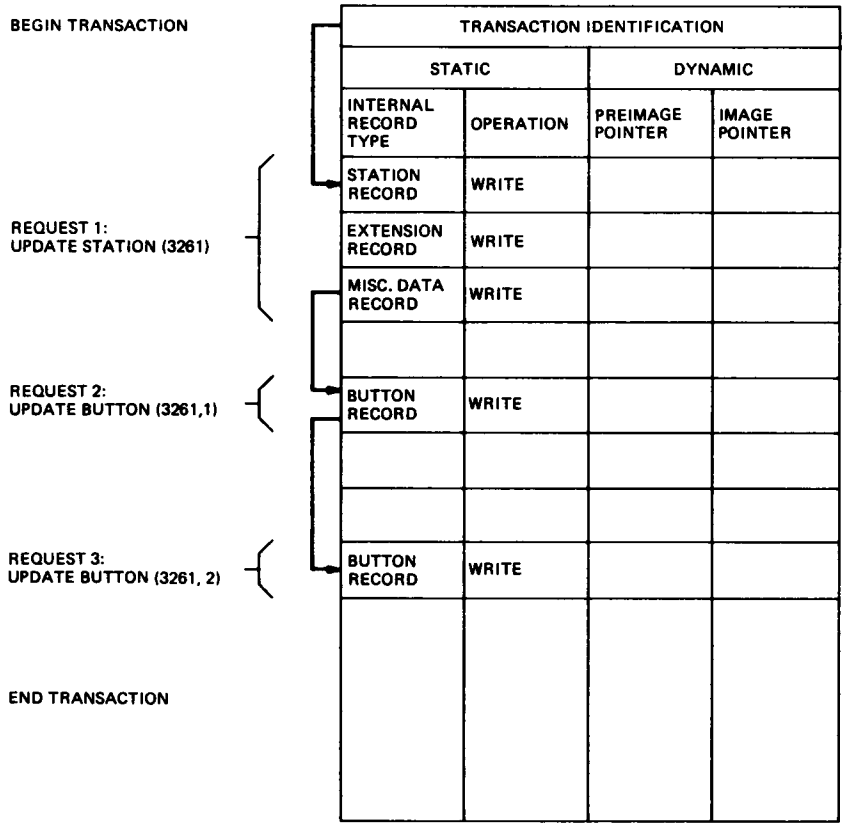

Fig. 7—Transaction execution table.

operations for this request (*object, action*) pair. The dynamic portion contains pointers to the preimages and images of the data to be changed. The preimages are used for possible recovery actions, i.e., for backing out of a transaction.

The instances of the *execution tables* (one per *command execution and validation* layer request) are stacked on behalf of a transaction. For each transaction in progress, the current pointer points to the last entry processed. When a transaction terminates successfully, all the tables are deallocated. The tables are used to perform transaction back-out and recovery.

3.4 Database validation

As we have seen, the *data access and storage* layer distributes system data to internal records, and this function is guaranteed to be robust because form transactions and a locking mechanism are utilized. To help simplify the software in the administration database access module, these functions are accomplished in a data-independent fashion wherever possible. The function of ensuring the correctness (i.e., semantics) of the database is accomplished in the administration database update/validation module.

Validation is accomplished as early as possible in the data entry process. Each field on a form is checked before the user is allowed to advance to the next field. This checking can be as simple as a range validation or as complex as checking that a station is installed before it may be entered into a hunt group, thus ensuring the correct operation of calls terminating to the hunt group. After all the necessary or desired information is typed on a form, the entire form is validated when the user depresses the *enter* key. At this time all interfield relationships are examined and verified. An example of this is ensuring that a multibutton station has at least two call-appearance buttons installed on it. Two buttons are necessary for the correct operation of the conference and transfer features.

IV. TABLE-DRIVEN SOFTWARE STRUCTURE

System management software in System 75 is a good example of the use of table-driven software. This table-driven approach is chosen for gains in memory usage and for ease of adding new administration objects. At each layer of system management software there are examples of modules which use this approach, such as the forms definition dictionary, administration database update/validation, and administration database access. To illustrate the approach, the software structure of the administration database access module, along with the tables that are used by this generic software, are briefly described below.

Generic software belongs to the following categories:

1. Table allocation
2. Request interpretation
3. Field conversion
4. Invocation of internal record library functions
5. Transaction control.

These five categories are executed, in the order shown, for any request from the command execution and validation layer. The precise software used is, for some categories, dependent upon the *action* (e.g., update or *add*) requested. However, this software is almost completely independent of the objects involved. Knowledge of object manipulation, conversion, and storage is contained in the following four major types of tables:

1. Request driver table—Contains all the actions that are valid for a particular object, along with pointers to other tables that are used for each request (*object, action*) pair.

2. Operation table—Provides a sequence of run-time operations to be performed on internal records for a given (*object, action*) pair. The order of execution is implicit in the order of operations in this table.

3. Field map table—Describes the field mapping between the individual fields of an object and the individual fields in associated internal records.

4. Internal record location table—Indicates which process contains each internal record.

An example of the use of these tables is shown for (station, update) in Fig. 8. Note that in the operation table the station record is listed first. This is critical for (station, add) since the internal identifier for the station is generated by the insertion of the station record. This internal identifier is needed in the other records associated with the station.

For some objects, the object-to-internal record mapping is data dependent, that is, the set of internal records that a particular object maps to depends on a particular transition of field values. These data-dependent transformations are handled using a *pseudo-object* concept. The basic idea is that *pseudo-objects* are used to enumerate all the different possible object-to-internal record mappings for a particular object. This enumeration enables the administration database access module to use the regular tables even for the data-dependent transformations. The *pseudo-object* concept is completely hidden from the higher-level software.

V. CONCLUSION

This article has presented two aspects of System 75 system management: the system access terminal and the layered software archi-

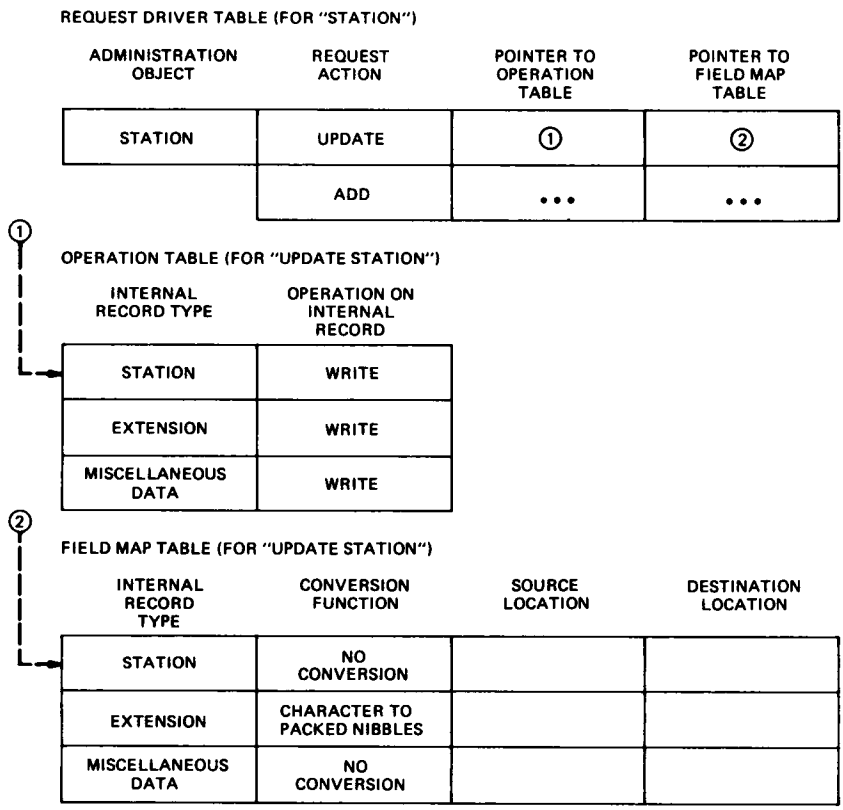

Fig. 8—Example of table-driven software.

tecture. The system access terminal provides a vehicle for the simple presentation of a complex database. The layered software architecture makes this all possible by providing data view mapping, database validation, transactions, and concurrency control. Moreover, it provides a structure that allows expansion to new functionality.

REFERENCES

1. L. A. Baxter et al., "System 75: Communications and Control Architecture," AT&T Tech. J., this issue.
2. W. Densmore et al., "System 75: Switch Services Software," AT&T Tech. J., this issue.
3. C. J. Date, *An Introduction to Data Base Systems*, 3rd edition, Reading, MA: Addison-Wesley, 1981.

AUTHORS

Anna S. Melamed, B.S.E. (Electronical Engineering), 1969, Warsaw Polytechnic; M.S.E. (Computer Information and Control Engineering), 1972, The

University of Michigan; M.S. (Mathematics), 1974, The University of Michigan; Ph.D. (Computer Information and Control Engineering), 1977, The University of Michigan; Bell Laboratories, 1978–1982; AT&T Information Systems Laboratories, 1983–1984; AT&T Bell Laboratories, 1984—. Ms. Melamed has been involved in the definition, design and implementation of various software systems such as a Test Language Parser, a database restructuring system for the AT&T 3B20-DMERT (Duplex Multiple Environment Real-Time) applications and system management for System 75. She is currently involved in performance analysis of a distributed computer system.

Gerald A. Reisner, B.S. (Applied Mathematics), 1967, New York University, School of Engineering; M.S. (Applied Mathematics), 1970, Adelphi University; Ph.D. (Mathematics), 1974, University of Minnesota; Hazeltine Corporation, 1968 to 1970; Bell Laboratories, 1974 to 1983; AT&T Information Systems Laboratories, 1983—. At Bell Laboratories, Mr. Reisner's initial assignment involved characterizing network blocking under various conditions, including retrials. He also provided performance measurement requirements for local switching systems. In 1978 he moved to software development for the Network Control Operations Support System, a minicomputer-based system used to install and maintain private networks. Since 1980 he has designed Switch Services Software for System 75, and since 1983 has been a Supervisor for System 75 Switch Services Software. Member, Tau Beta Pi, ACM and MAA.

Harold K. Woodland, B.S. (Physics), 1971, Morgan State University; M.S. (Engineering), 1973, Cornell University; Bell Laboratories, 1973–1983; AT&T Information Systems Laboratories, 1983—. At Bell Laboratories, Mr. Woodland's work has included design of special trunk circuits and software development for the *Horizon*® communication system Customer Access Unit, the Enhanced 911 Emergency Reporting System (E911), the *Horizon* ACD and System 75 System Management. Since 1981, he has been Supervisor of the System 75 System Management and Administration group.