Samhita—A data base management system

Naveen Prakash, P. Sadanandan, S. M. Desai, K. Rama Rao* and P. N. Gejji

National Centre for Software Development and Computing Techniques, Tata Institute of
Fundamental Research, Homi Bhabha Road, Bombay-400 005

In this paper, a DBMS called Samhita is described. Samhita is based on the network model of data.
The prime motivation behind Samhita was to re-examine the notion of currency indicators with a
view to minimising side effects in currency setting and to introduce the concept of multiple position-
ing in the data base. An attempt was made to impose a discipline on data base definition by
restricting the network model itself. It is expected that as a result of these efforts a methodology
for programming for data bases will emerge.

(Received March 1979)

This paper describes a data base management system called Samhita which is based on the network model of data. It
is being coded in BLISS-10 and is expected to be made available on the DEC System 10 at the NCSDCT shortly. The prime
motivation behind Samhita is to re-examine the concept of currency indicators as propounded by the DBTG(CODASYL,
69; CODASYL, 71), to introduce the notion of multiple positioning in the data base and to help in developing a
methodology of programming for data bases.

In the following, we list the special features of Samhita.

1. In order to impose a discipline on data base systems the
   definition of a network as given by Date (1976) was accepted.
   Specifically, Samhita puts the following restrictions on the
   network model:

   (a) Sets cannot contain more than one member record type.
   (b) Areas Realms (CODASYL, 1969; 1971) are not
       supported.
   (c) No repeating groups are allowed.

2. The data base key has been replaced by an internal system of
   pointers. These pointers are relative addresses of record
   occurrences within a given record type. It is not possible for
   the user to access these pointers since no 'system
   communication locations' for accessing them are defined by the
   DBMS.

3. In the design of the Data Manipulation Language we have
   tried to ensure that the static behaviour of the program
   reflects its dynamic behaviour. This implies that, ideally, we
   would not tolerate the presence of side effects in currency
   setting. However, we have fallen short of this ideal when
   treating the STORE and DELETE statements. The reasons
   for this are given later in the paper.

4. The DML has been augmented by a statement which allows
   the programmer to retain his position in a set or record and
   to return to it at a later point.

The organisation of the paper is as follows: First, we discuss the
method adopted in Samhita for multiple positioning in the data
base. The side effect problem is presented next. Lastly, the
DML of Samhita which minimises this problem is described.

1. Multiple positioning

According to the DBTG proposals there are four kinds of
 currencies; namely, run unit, record, set and area. The currency
indicators essentially maintain various positions in the data
base. If a user wants to maintain multiple positions within a
given record type for example, he can save the currency
indicator concerned in a location defined by him and restore it
before use.

*Computer Group, ECIL, Hyderabad 500 762, INDIA.

In Samhita, all currency information is kept in system
maintained tables. These tables cannot be accessed by a user
directly. The user can maintain multiple positions in the data
base by invoking the MARK/RESTORE facility available in
the DML of Samhita. When a user issues a MARK, he specifies
the currencies to be stored and tags the currency with a name
thus enabling him to return to the status referred to by the
name. Samhita saves the currencies in a system defined table.
It maintains a separate table which contains the tag specified
by the user and an index into the table in which the currencies
have been stored. The RESTORE supplies the tag to which the
data base status is to be restored. The system accesses the table
which contains the currency indicators and establishes these as
the current of record type and/or the set type concerned.

It can thus be seen that the internal system of pointers in
Samhita is totally safeguarded from the user. Since there are no
system communication locations for currency indicators
assigned by the DBMS, a user can never access these indicators.
Further, in case he wants to maintain multiple positions in the
data base, the identifier which he supplies in the MARK
statement acts as a tag which identifies a particular position in
the data base which is to be saved. The identifier supplied is not
allocated space and thus does not contain a value of any
record pointer.

In this way, the record pointers are kept totally safeguarded
from the user. It is possible to maintain multiple positions in
the data base without compromising protection of record
pointers. It should be noted that, even though not implemented
by us, the MARK/RESTORE facility could be extended to
allow currency lists similar to CODASYL keep-lists to reduce
the number of tags.

2. Side effects in currency setting

In the DBTG proposals side effects in currency setting are
allowed. There are two sources of these side effects; namely
FIND and STORE. We will now examine the nature of these
side effects for both these statements.

2.1 Side effects in the FIND

Let us consider the rules for currency setting for FIND. All
versions of FIND (corresponding to various Record Selection
Expressions) establish the record selected as the current of
run unit.

In addition, if the SUPPRESS clause is absent, then the
record selected becomes the current of its record type, current
of its area and current of all those set types in which it partici-
ates either as an owner or as a member.

Essentially the FIND statements allow for content address-
ability and also provide a navigational facility within the data

THE COMPUTER JOURNAL, VOL. 24, NO. 1, 1981
CCC-0010-4620/81/0024-0048 $02.00 © Heyden & Son Ltd, 1981
base. Side effects in currency setting are possible in both classes of the FIND.

Case 1: Navigational FINDS
While navigating within a set, one can select, for example, the next record in the set. However, by the rules which establish currency, currencies of other sets may change if the record selected participates in these. The programmer may not have intended to cause these changes. Further, it is not even clear from the FIND NEXT RECORD OF set-name SET statement that such a change has occurred. Clearly a side effect has been produced.

Case 2: The content addressable FINDS
Let us consider the statement

\[
\text{FIND NEXT DUPLICATE WITHIN record-name RECORD;}
\]

According to the currency rules of the DBTG, the record selected by this statement becomes the current of all sets in which it participates either as an owner or as a member. It is again apparent that the text of the statement does not reflect the currency changes in the sets concerned.

2.2. Side effects in STORE
Side effects can be caused by the STORE statement as well. This statement establishes the record stored as the current of the run unit. In addition, if the SUPPRESS clause is absent, it also becomes the current of its record type, the current of its area and the current of all those set types in which it participates either as owner or as automatic member.

The major cause of these side effects is the SET OCCURRENCE SELECTION clause. There are two options available to the user in the specification of SET OCCURRENCE SELECTION, namely, THRU CURRENT OF SET and THRU LOCATION MODE OF OWNER. It is the latter which is responsible for causing side effects because it allows the storage of records into set occurrences other than the current occurrences of these set types.

There are two methods by which side effects can be avoided in the STORE statement.

(a) Do not allow AUTOMATIC membership; this implies that the STORE operation will acquire space and put the values of the various data items of the record in the user work area into the space acquired. If set participation for this record is to be established then this should be done procedurally.

(b) The second option is to retain the AUTOMATIC membership option but to allow set occurrence selection THRU CURRENT OF SET only.

We feel that the first alternative is highly restrictive and have, therefore, not accepted it. The second alternative can be used in the following manner:

\[
\text{STORE record IN set list;}
\]

where 'set list' is the list of all those set types in which 'record' occurs as an automatic member. This makes currency setting explicit. It is debatable, however, whether this defeats the purpose of an AUTOMATIC membership clause or not. We, for our part, have decided to allow side effects in the store operation by not asking for the list.

3. Data manipulation language of Samhita
The data manipulation language supported by Samhita provides all navigational and content addressing facilities provided by the DBTG. However, it does so in a controlled manner. This discipline in the DML arises from our decision to minimise side effects in currency establishment. Such a decision implies that the DML statements should reflect any currency changes that may occur upon their execution. We have tried to ensure this. Further, set occurrence currencies cannot be established 'behind the scenes'. An ESTABLISH statement has been introduced which must be used to identify set occurrences. No DML operation on a set is allowed until an occurrence of the set has been established. Since Samhita does not support areas, there is no currency indicator for areas. We have also decided to do away with the current of run unit. As will be seen later, all DML statements operate on either the current of record type or the current of set type thus rendering the concept of 'current of run unit' superfluous. Therefore, in Samhita, currencies are maintained only for record type and set type.

In order to study the effect of these decisions, we give the DML specifications for Samhita as follows:

1. \[
\text{\{FIND \} record-name FOR identifier;}
\]

For this statement to carry meaning it is necessary that the location mode of record-name be either 'indexed' or 'calc'. According to Samhita, indexed refers to the indexed sequential access method and calc to a hash procedure which may be either system or user defined. (All hash procedures are maintained in a library). The identifier must contain the key value to be used to select an occurrence of record-name.

This statement allows content addressability in the data base. It establishes a record position in the data base and with the OBTAIN option brings the selected record into the UWA (user work area). This statement does not assume an initial position in the record area but in both cases, it sets the current of record to the record selected. The syntax does not specify establishment of any set currency. Thus no set currency is established, nor is any set currency updated by the execution of this statement.

2. \[
\text{ESTABLISH record-name IN set list;}
\]

This statement assumes that a record occurrence of the record type specified has been selected. Set occurrences are established for all the sets specified in the set list by identifying those occurrences where the current of record-name participates either as owner or as member. Currency of each of these sets may therefore be set either at the owner type or member type depending on the participation of the specified record in the set concerned. Furthermore, navigations within these sets will be based on the positions thus established.

3. \[
\text{\{FIND \} \{FIRST LAST \} \{NEXTPRIOR \} \{record-name \} \{IN set-name \};}
\]

In the record-name option of this statement, the following holds:

(a) The FIRST and LAST specifications do not assume an initial position in the record type. FIRST and LAST refer to the physical first and physical last records respectively.

(b) The NEXT and PRIOR specifications assume that a position has already been established in the record type. The next or prior record is found with respect to this position. As before, NEXT and PRIOR refer to the physical next and physical prior records respectively.

In both cases (a) and (b) above, a record occurrence is selected and is established as the current of its record type. No set currencies get modified.

The set name option of this statement assumes that an initial position in the set type has been established. The specified record occurrence is selected and it becomes the current of its record type as well as the current of the set type.
4. \( \{ \text{FIND} \} \text{ OWNER IN set-name 1 OF CURRENT OF record-name in set-name 2} \);  
An initial position is assumed in both the record and set alternatives. The former alternative selects the owner in set-name 1 and makes it the current of its record type as well as the current of set-name 1. The latter alternative selects the owner in set-name 1 for the record which is the current of set-name 2. Both the current of record-name and current of set-name 1 are set to this record. The currency of set-name 2 is unaffected. The case when the optional part is omitted is a special case where set-name 2 is assumed to be equal to set-name 1. By virtue of the fact that this statement selects an owner record it establishes an occurrence of a set.

5. \( \{ \text{FIND} \} \text{ DUPLICATE OF record-name}; \)  
This statement assumes an initial position in record-name. The next physical record having the same key value (as defined earlier) as the current of record-name is found (and 'obtained' if desired) and the current of record is modified to this new position. No set currencies are affected by this statement.

6. \( \text{OBTAIN record-name}; \)  
This statement brings the current of record-name into the UWA for the record, but does not change any currency value.

7. \( \text{INSERT record-name \{ INTO set-list \ INTO ALL-SETS \} }; \)  
The INSERT statement operates on the current of record-name and makes it the current of those sets into which it has been inserted. The target set types can either be explicitly listed in the INTO part of the statement or the insertion could be specified to take place in ALL SETS in which the record concerned is a participant. The set occurrence(s) in which insertion takes place must have been selected before the INSERT is performed.

8. \( \text{REMOVE record-name \{ FROM set-list \ FROM ALL SETS \} }; \)  
This statement operates on the current of record-name and removes it either from all the sets which have been explicitly named in the set list or from ALL SETS in which the record-name participates as a member. Note that the record-name must be an optional member of sets in order that the REMOVE be effective and that the set occurrence(s) from which removal takes place must have been preselected. After the record concerned has been removed the currency of the set is affected according to the following rule:
If the current of record-name is the same as the current of the set from which removal occurs then the current of the set is changed to point to a 'gap' record occurrence—the gap created by the removal of the record from the set occurrence. If however, the current of record-name is different from the current of the set then after removal of the record no change is made to the set currency. In both the cases considered by this rule, the current of record-name is unaffected by the remove operation. As a result of this rule the user can continue to navigate through the set.
It must be noted that if the REMOVE were to be defined to operate on the current of set then the currency setting rule would be simplified. However, such a decision would decrease the flexibility of the construct. Hence, our decision.

9. \( \{ \text{DELETE} \} \text{ record-name \{ ONLY SELECTIVE \} }; \)  
The DELETE operates on the current of record-name and after its execution the current of record-name points to a gap occurrence. As in the case of the REMOVE this enables the user to find the next record of the same type in the data base subsequently. Since the various options of the DELETE are defined in the same manner as in the CODASYL proposals, side effects can occur as a result of the DELETE operation. It is not clear (to the designers of Samhita) how these side effects can be avoided.

10. \( \text{MODIFY record-name}; \)  
This statement operates on the current of record-name. Record currency is not altered. The present version of Samhita does not allow modification of key fields.

11. \( \text{STORE record-name}; \)  
This statement causes the record-name available in its UWA to be written out in the data base. It further inserts this record occurrence into all the sets in which it participates either as an owner or as an automatic member. In the case when the record occurrence participates as an automatic member in some set type then a set occurrence of the set type concerned must have been established before the STORE operation. The system does not entertain the STORE request if this has not been done. In the case when the record occurrence to be inserted is an owner of a set, then a set occurrence with no members is established. After execution of the STORE the current of the sets concerned points to the record just stored.
It should be carefully noted that when an owner is stored in the data base, a set occurrence is established. However, when storing an automatic member, the set occurrence must have been established beforehand.

12. \( \{ \text{MARK} \} \text{ record-name \{ ALL \} FOR set-name WITH identifier}; \)  
The MARK facility, along with the companion RESTORE facility, allows the programmer to mark positions in the data base which are likely to be required again. As discussed elsewhere, the explicit use of data base keys and SUPPRESS CURRENCY options of the DBTG model are eliminated in Samhita and the MARK/RESTORE facility is provided to achieve the same capabilities in a more disciplined manner. The MARK statement functions as follows:
(a) In the case of the record option, the current record occurrence is saved with the identifier specified. The identifier can thereafter be used to return to the present position of the record subsequently, if desired.
(b) In the case of the set-name option without the ALL clause, the current of the set type is saved with the identifier for the later restoration of the set status.
(c) In the case of the set-name option with the ALL clause, apart from the current of the set specified, the current of the owner record and the current of the member record are also saved with the identifier.

13. \( \text{RESTORE identifier}; \)  
This statement restores the currency as determined by the identifier. MARK will have classified the identifiers according to the three functions outlined above. RESTORE therefore does not require the class to be repeated in the syntax. The result of RESTORE will be a record currency, a set currency or a set currency with the owner and member currencies depending on the identifier.
4. Conclusion
The Database Management System Samhita makes an attempt to enforce some discipline on the data base user. This is done in the following directions:
(a) The network structure envisaged makes the data base more intellectually manageable
(b) The Data Manipulation Language minimises side effects in handling record and set currency
(c) The concept of the data base key has been replaced by that of an internal system of pointers inaccessible to the user. At the same time, it is possible for the user to maintain multiple positions in the data base without tampering with record pointers. A potential source of errors is thus eliminated from the Data Manipulation Language.

References

Book review

This collection of ANSI FORTRAN programs in digital signal processing represents the fruits of a four-year project undertaken by a distinguished group of specialists active in the field. A broad spectrum of programs is included covering both applications and design techniques. Each program has been subjected to comprehensive tests by reviewers located at sites in the USA, Canada, Mexico, UK and Continental Europe using the programs on a variety of machines and necessarily a variety of FORTRAN compilers. To facilitate this all programs are written in ANSI PFORT and all machine dependent quantities are specified in the PORT library. Values required for use in the function subprograms related to a large number of machine types (all American) are given in an Appendix.

The editors and authors have given particular attention to uniformity and standardisation of programming method. For example, the standardisation of format by renumbering all statement labels by inserting CONTINUE statements to force every DO statement to end on a unique CONTINUE statement and by systematically spacing and indenting each statement in a program. All programs are liberally and clearly commented with an initial statement describing the function, with bibliographical references if required, and further COMMENT statements throughout the program. There should be no difficulty in following the logic and path structure of any of these programs. This is more than a mere collection of 'other people's programs' however. Each chapter includes its own introduction and review of methods, providing a readable tutorial in signal processing techniques. Good bibliographies are given which include key papers in each subject.

Of particular value are the set of Fast Fourier Transform routines selected and edited by no less an authority than J. W. Cooley. Several of these have not previously been published, although all have been most thoroughly tested and edited. Some of the FFT programs are very fast, obtained at the expense of instruction memory and programming complexity. Others take advantage of symmetries in the data to save computation. Mixed radix programs and optimal mass storage routines are included. Sizing, timing and accuracy of these FFT's have not been neglected and are well summarised in tables included at the end of this chapter.

Only four programs are included in the chapter on Power Spectrum Analysis from the many that have appeared in earlier IEEE publications. However, the four chosen cover the periodogram, correlated, coherence and cross-spectral estimation and are sufficiently flexible for most requirements. All are well documented and the excellent introduction gives several examples of use.

The chapters on high speed convolution, FIR and IIR digital filtering will be much thumbed by those with filtering problems—and this applies to everyone who has digital signals to process! The high speed convolution program utilises the well known overlap—add technique due to Stockham. The most important of the FIR filter programs uses the Remetz exchange algorithm, which has wide use. Others use classical methods including the windowed linear phase filter permitting program choice of no less than seven windows. Programs chosen for inclusion in the chapter dealing with IIR filters reflect the general design problem of nonlinearity in the filter coefficients compared with the linear FIR filter case. There has been considerable publication of IIR program designs in the last decade, many having particular advantages in specific cases. The filter programs included here are general purpose programs representing ease of use rather than fast operation. This is probably optimum for the type of researcher who will use this book. The programs are among the longest documented programs included in the book, but are designed to provide much information on performance, e.g. noise figures, as well as allowing optimum design parameters to be selected for the reader's application. It is surprising to find routines included to permit plotting of filter characteristics on the line printer since few scientific installations fail to contain better graph plotting devices.

The remaining chapters contain much useful information as well as programs on Linear Prediction Analysis of Speech Signals, Cepstral Analysis and Interpolation and Decimation. This last contains a useful set of programs on the fundamental operations required in digital signal processing which are sufficiently well thought out and documented to deter any researcher from writing his own routines. It is to be expected with a collective work of this size, taking several years to produce, that some omissions will be felt, particularly as one would wish to see other topics treated with the same thoroughness and attention to standards. It would be nice to find included sets of programs for Walsh and related transform analysis, spectral analysis by Pisarenko maximum entropy methods and some general programs on 2-dimensional analysis (although Singleton's 2-dimensional transform program is included). The only regrettable omission in the reviewer's view is the absence of IIR filter programs using the Bi linear Z-transform which enables almost any continuous filter function to realise its digital equivalent.

In summary this book represents a remarkable and valuable addition to the signal processors' library, produced at an extremely reasonable cost for the amount of effort and thought that has gone into its preparation. The members of the Digital Signal Processing Committee of the IEEE Acoustics, Speech and Signal Processing Society are to be congratulated on a fine result which will make the task of processing digital signals considerably easier for many in the years to come.

K. G. Beauchamp (Lancaster)