Def Sci J, Vol 38, No. 2, April 1988, pp 141-159

Database Model for Military Operational Information System

Sarbjit Singh

Directorate General of Signals, Army HQ, New Delhi-110 011

ABSTRACT

The existing system of exchange of battlefield information among Commanders at various levels is not responsive to the needs of today's fast changing battlefield scenario. This paper proposes a network of field computer systems with hierarchical database which includes organic data of various units as well as battlefield activity data.

1. INTRODUCTION

The modern warfare involving extenstive mobility and improved weapon systems of extended range and accuracy demands that a field force commander should be able to re-adjust his plans to varying tactical situations. For a commander who is to make accurate and efficient plan, the flow of battlefield information must be timely, accurate and relevant. This necessitates that data must be quickly captured at the source and processed efficiently so that appropriate information is disseminated to the required levels of decision making. The existing system of battlefield information management in the Armed Forces was evolved during the Second World War or before. Perhaps, it was suitable for the condition which then existed. In the existing telegraphic or telephonic messages in text form, which is transmitted over lines/radio communication circuits, the processing of information is generally manual where storage/retrieval of information follows conventional filing approach. It is therefore essential to take a fresh look at our information handling technique and exploit full potential of real-time computer system and data communication networks.

1.1 Real-Time System

The ability of a computer system to store and process large volume of data is the most important ingredient for timely and accurate decision making. It might be a question of generating an instantaneous control signal in a chemical process, or

Received 1 July 1987

interactive dialog to retrieve information and react in battlefield environment. Any wrong decision made based on intuition and inadequate or inaccurate information might mean economic loss of equipment, material, human lives and defeat in battlefield environment. During military operations, intelligence about 'own' and enemy has always been the pre-requisite for successful plan. The present day mobility and weaponary has given new dimension to the warfare, where real-time computer system can be the most powerful tool for a field force commander. Its ability to handle volumes of data and provide vital information in an interactive mode can change the likely outcome in a battlefield. Since the time is generally at premium, the data must be so structured as to ensure fast response. Obviously, old concept of organising data in flat file format requiring repeated SCAN, SORT and MERGE operations for every query is inefficient. As such, there is a need to develop an operational information data base structure to provide real-time information retrieval.

1.2 Tactical Command and Control

In any future war, the forces involved would use their total resources, and need accurate decisions at various echelons. The unique characteristics of effective command and control system are speed and accuracy. A representative model of tactical command and control cycle is shown at Fig. 1 where activities like collection of information, transmission of data, processing of information, dissemination of information and various actions take place continuously. It is therefore essential that real-time computers are utilised as an aid for tactical command and control functions.

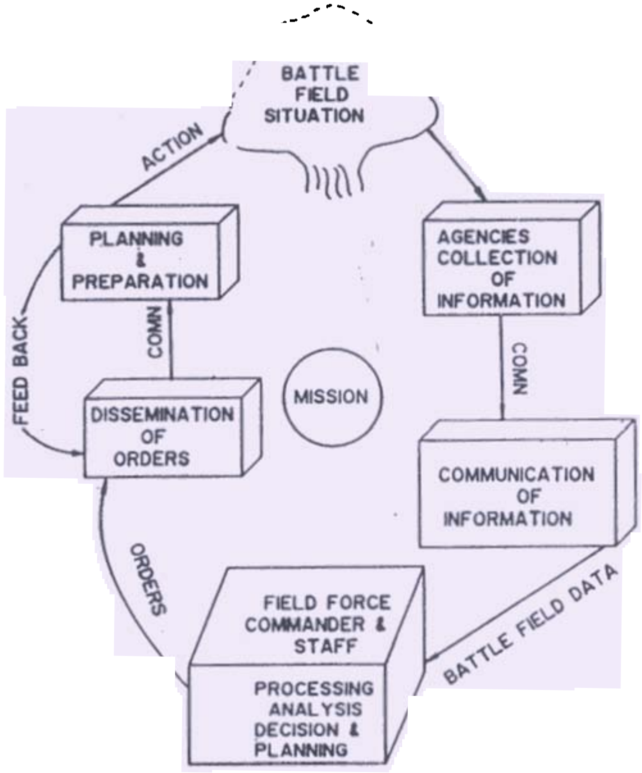

BATTLE FIELD SITUATION
ACTION
PLANNING & PREPARATION
AGENCIES COLLECTION OF INFORMATION
FEED BACK
COMN
MISSION
COMN
DISSEMINATION OF ORDERS
COMMUNICATION OF INFORMATION
ORDERS
FIELD FORCE COMMANDER & STAFF
BATTLE FIELD DATA
PROCESSING ANALYSIS DECISION & PLANNING

Figure 1 Tactical command and control cycle

1.3 Reaction Time

The reaction time in the battlefield is of decisive importance. Reaction time can be improved if information response time is minimal. In order to improve reaction time, database system must be responsive to the user requirements. Some of the factors which help in improving reaction time are given below :

(i) Fast means of data collection and transmission.

(ii) Fast,accurate and reliable processing of large volumes of data.

(iii) 'Single source, single transaction' concept of updating database.

(iv) Automatic indexing, filing and storage of data.

(v) Clear and differentiated presentation of information relevant to the level of HQ/staff.

(vi) Automatic plan for evaluation of alternatives and decision making.

(vii) Timely dissemination of orders.

(viii) Automatic feed-back and monitoring.

(ix) Periodic dumping of information and fall-back procedure to reduce vulnerability.

(x) Fast access to database with shortest response time.

(xi) Effective security and integrity.

2. NEED FOR DATABASE

The importance of quick reaction time necessitates efficient data organisation. However, use of traditional files does not meet the vital requirement of providing up-to-date, accurate and timely information for correct and proper decision making. It is indeed the power of real-time DBMS and data communication networks which facilitates information flow within acceptable response time limits.

2.1 Drawbacks of Conventional Data File Environment

(i) Data redundancy.

(ii) Data integrity problems such as insufficient security of stored data, inadequate recovery procedure in case of failure, inflexibility to changes, higher programming and maintenance costs, and difficulty in managing procedure of computer operations.

(iii) Limited data sharing.

(iv) Data availability constraints.

(v) Difficulty in management and functional control.

2.2 Database Concepts

Database can be defined as a collection of inter-related stored-together data with controlled redundancy to serve one or more applications in an optimal fashion. The data structure in a database is independent of the applications. This implies that the database contains data for many users where each user will be concerned with a portion of the data. However, individual pieces of data may be shared by different users. Updating and retrieving of existing data in a database follow a common and controlled approach. Its salient features are :

(i) Data must be organised in such a way that information can be obtained as soon as required, and in any sequence or format.

(ii) The user may pose a wide variety of queries about the data stored in the database. Since it is not possible to anticipate in detail the nature of all possible queries, there is a requirement to have fast and flexible search capacity using interactive query language.

(iii) In on-line system, queries from different users cannot be batched since they occur randomly. It is therefore necessary that data should be so structured as to give fast response to all queries.

3. QUERY LANGUAGES

3.1 Need

In a battlefield environment, it is essential to enable common users of the system, who are not application programmers to query the database and obtain the required information in an interactive mode. To meet this requirement a suitable choice of a query language should be made from a wide variety of database query languages available.

3.2 Features of Query Language

In view of the fast response time required for operational information, the language must be designed for on-line interrogation. An online user must have capability to carry on dialogue with the system in an interactive mode to specify complex queries and step-by-step narrow down the search until the required information is obtained. Some of the essential features are:

(i) *Simplicity :* Query language should be simple to use where a non-specialist user can make query and extract required information.

(ii) *Non-procedurability :* User should be able to specify what process the system should perform rather than how to perform.

(iii) *Completeness* : All relevant information should be accessible by the query language.

(iv) *Extendability:* To allow future expansion of data structure.

(v) *Data independence :* To allow the user to be free from data structuring of the database.

4. REQUIREMENTS

4.1 Typical User Requirements

To design an efficient database structure and information retrieval system for providing a real-time operational information for field Army, the system must provide the following facilities :

(i) Organisational structure of a higher head quarter (HQ) to include order of battle (ORBAT) upto unit level.

(ii) A middle HQ (MHQ) or lower HQ(LHQ) may have two or upto four sub-formations/units.

(iii) Remote data entry/retrieval upto unit level.

(iv) Security of database to allow only authorised access.

(v) Validation of all data carried out before updating database and error messages displayed accordingly.

(vi) Giving indication of inventory item falling below a given scale of establishment.

(vii) Each unit to have one inventory record giving vital items like tanks, guns, vehicles, radio sets and personnel. Each unit will also have number of activity records, logged in chronological order, giving details of various activities in their area of operational responsibility.

(viii) System should be able to give answers to various queries relating operational information in real-time.

4.2 Typical Technical Requirements

The system design and implementation must have the following features :

(i) Program structure to be modular and follow well defined structured coding techniques of PASCAL.

(ii) All inputs to the system must be generalised for handling any size of database structure.

(iii) Design of directories for activity code, activity subject code and passwords. The system must also have reinforcement level for inventory, arrival of transactions (messages) to be in any order and their processing to be on-line batch mode (simulating distributed data terminals concept of real-time, check overhead (OVH) factor for processing individual transaction, check response time for the user to ascertain real-time functioning, and it must provide a generalised query interpreter in PASCAL to give fast response to the various queries.

5. PROPOSED SYSTEM

5.1 Choice of Database Structure

Three most commonly used database structures are network, relational and hierarchical. However, implementation of all these techniques is still in infancy in our country. Most organisations in the Defence Services follow hierarchical structure inherently as shown in Fig. 2. Besides this factor, ease of implementation has led to the choice of hierarchical database structure.

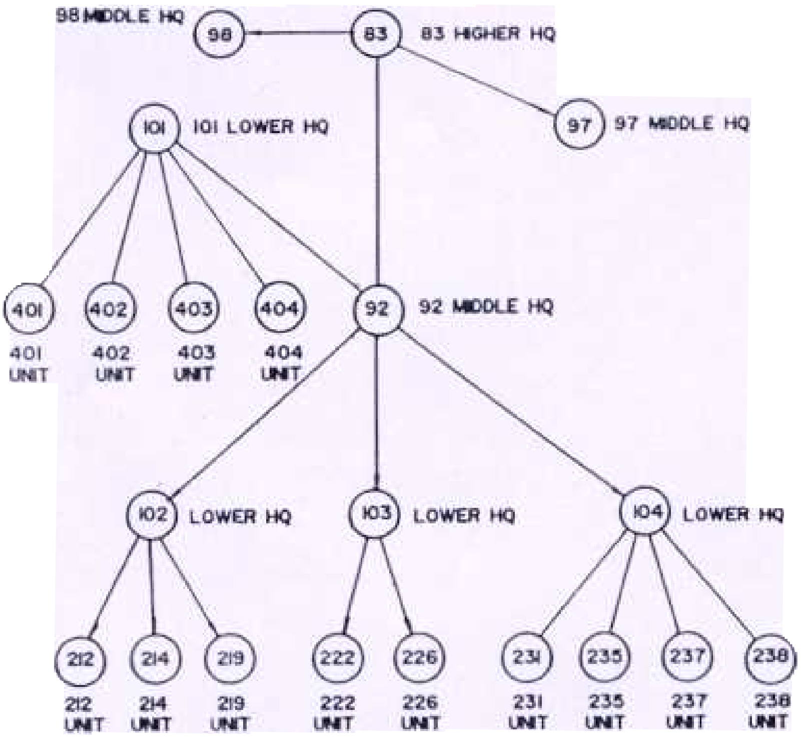


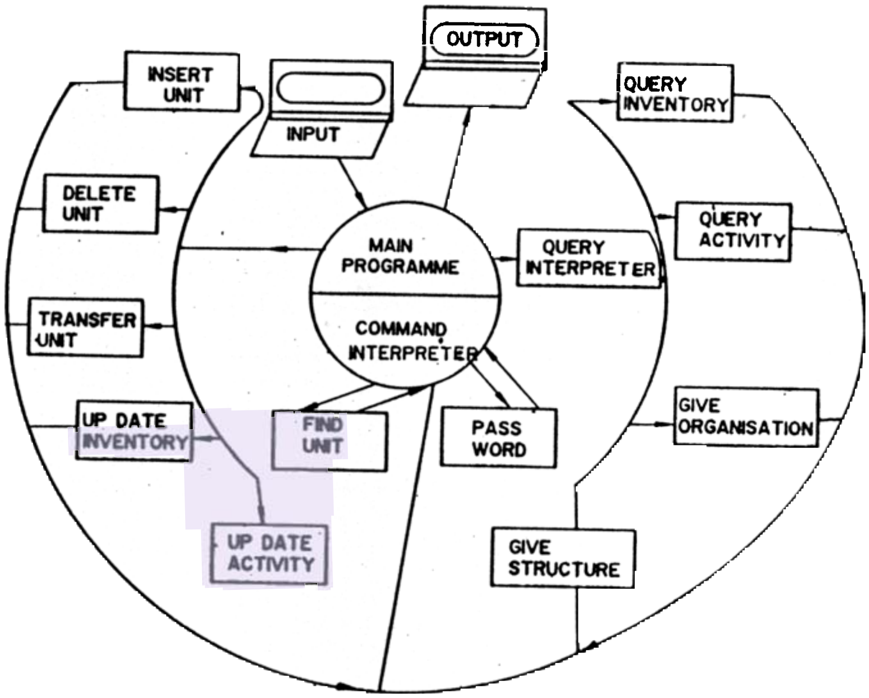
Figure 2. Hierarchical data structure

5.2 Choice of Query Language

Choice of an efficient query language is related to databse structure. However, for this project, PASCAL based query language has been evolved which is simple to use and meets the requirements spelt out earlier.

5.3 Database Design

The database system has been designed for building competence in implementing various techniques in database management, and evolving methodology for real-time application in the military operational environments. It has been implemented using structured programming techniques in PASCAL on DEC-10 Computer System at NCSDCT. The database system has been modularised into twelve easily readable and maintainable modules using concepts of pointers and recursive techniques. The system flow and level of inter module communication is shown in Fig. 3.



5.4 Type and Frequency of Transaction

In real life situation all transactions, whether initial creation or maintenance of database, or data retrieval will be originated and transmitted from remote data terminals. The occurrence of these transactions will be in random order. However, in order to simulate an equivalent behaviour in the present implementation, all types of transactions are placed in a queue in random order. Once the system is invoked, these transactions are processed sequentially.

5.5 Security of Data

Importance of security of database, particularly in Defence environment, cannot be over emphasised. Measures for protection of database against unauthorised access, alteration or deletion must be enforced. However, for this project, only some salient features of security have been implemented by introducing 'Branch Codes' and 'Passwords'. In this scheme one letter code for each 'Branch' coupled with four letters 'Password' has been devised for each branch at various HQ. The directory of passwords (PASDIR) is stored in the system and is changed/updated as and when required. In a practical situation the user will establish his identity during initial dialog with the machine and depending on the type of transactions he intends to process, he will be given specific access. A similar security system has been simulated where for every transaction, a password must appear.

5.6 Information Flow

Battlefield information system primarily consists of three basic elements namely; activity data, ORBAT status data, inventory status data. Since the inherent military structure for decision making follows hierarchical organisation, ORBAT data has been organised as hierarchical 'Tree' structure, where unit (leaf node) has one inventory data record and none or many activity records associated with it. These types of data records have been discussed in succeeding paragraphs.

5.7 Activity Data

Information regarding various operations in a sector/area are reported through messages over digital communication network. All valid messages are received from remote data terminals and are logged in chronological order for each unit. Depending upon the nature of the message (transaction), corresponding data is updated in the database and for certain critical actions special message can be flashed for drawing attention of a particular appointment. The user while originating a message also transmits manual assessment of a situation in a plain text form which forms part of the message. For this purpose, identification of the individual department at various HQ is indicated in the body of the message. All logged messages are processed for creating 'Historical File' while unconfirmed messages are dropped after a week. Likewise, all information pertaining to the activity of a unit is avoidable for a month. Example of some of these activities are :

(i) Movement of troops, vehicles, tanks, guns, etc.

(ii) Firing, shelling or bombing by a weapon.

(iii) Laying/breaching of mines.

(iv) Construction of bridges/roads.

(v) Destruction of tanks.

(vi) Concentration of armour.

In order to minimise verbosity in a message, two directories, activity code – 'ACTDIR', and activity subject code – 'SUBDIR' are maintained on disc.

5.8 Activity Code

A three-letter mnemonic code for representing various actions in the battlefield has been developed. This code is used in conjunction with another three-letter code giving activity subject. The main characteristic of the code is that it minimises verbosity of expression and still it is easy to pronounce and remember. It has special significance in battlefield environment where numeric codes are difficult to remember and are less reliable particularly, when a user is operating under stress and strain. The directory of these mnemonics is maintained on a disc file and is kept progressively updated with new codes. However, all additions/deletions of codes must be approved by the Corps HQ and disseminated to all concerned. Some examples of combined code for activity and its subject are :

(MOVTPS) : Movement of troops.

(CONATY) : Concentration of artillery

Sample list of activity code 'ACTDIR' and activity subject code 'SUBDIR' are given at Appendices A and B respectively.

5.9 ORBAT Data

Depending upon the tactical scenario field force commander will often switch forces from one location/axis to another to meet various contingencies. Likewise, required reinforcement in terms of additional units may come in at the discretion of force commander who is empowered to reconfigure any task force. As such, the system should cater for the following transactions :

(i) Affiliation of new units.

(ii) Deaffiliation of present units.

(iii) Transfer of units within a sector.

(iv) The extension of the present system can cater for maintaining 'movement history' for various units.

5.10 Inventory Data

The battleworthiness of a unit is assessed from its standard of training, morale and logistics. Besides other preparedness, all vital inventory must be held more than 80 per cent of the authorised scale. However, during war, wastage due to enemy action or normal breakdown are likely to occur considerably and time to time status of inventory must be known. This will help a decision maker to ensure timely reinforcement at a required level. The authorised scale for various types of units is held on disc file. Whenever an inventory update transaction is processed, the reinforcement level is checked and pointer to that effect is flagged for the decision maker. Unlike activity data which is local to an area of operation, the inventory data forms an integral part of a unit and it moves with the unit. As a representative sample, five inventory items namely, tanks, guns, vehicles, radio sets and personnel have been considered.

5.11 Database Organisation

The hierarchical 'Tree' structure of 83 higher HQ (field force with hypothetical ORBAT) as shown in Fig. 2 has been implemented. The database structure has the following structural parameters :

(i) 83 higher HQ is the highest formation and forms the 'Root' of the 'Tree'.

(ii) By using pointers and recursive techniques of PASCAL the 'Root' can have one to any number of 'nodes' at any number of levels. For purpose of implementation the 'Tree' has three MHQs as first level 'nodes' and each MHQ has four LHQs as second level 'nodes'.

(iii) Each LHQ (second level 'node') has a unit as a leaf (terminal) of the 'Tree'. However, if company/battery/squadron level data is also needed, each unit can have one to four leaves.

(iv) Each leaf has one inventory record which may have any number of items associated with it. For above implementation, five elements have been considered as representative sample.

(v) Each leaf may not have any action record or may have many such records logged in chronological order. Action record has data items like data, time activity, activity subject, activity quantity and text.

(vi) 'Nodes' (MHQ and LHQ) do not have inventory or action records associated with them. However, accumulated holding of their ORBAT can be computed if required.

(vii) Units (leaves) can migrate to any other LHQ. However, while data moves along with the unit, action record is created freshly as per activity in its new area of responsibility.

(iii) Whenever a node moves, all its successors also move.

6. PROGRAM MODULES

The system flow and interaction of different modules of the system are shown in Fig. 3. Each module is described in four parts giving their application, input, output and algorithm. As a sample, one of the module (main program) has been described fully in succeeding paragraphs while other modules have only been explained briefly without giving details of input, output and algorithm.

6.1 Main Program

6.1.1 Application

It is the main module which acts as a command interpretor. Initially it reads in the following four directories:

(i) PASDIR – Directory of passwords.

(ii) ACTDIR – Directory of activity codes.

(iii) SUBDIR – Directory of subject codes.

(iv) INVDIR – Directory of 80 per cent inventory levels.

It invokes procedure PASSWORD to check validity of the user. In case there is an error in the user's password, current transaction is aborted and next transaction is taken for processing. However, if password is 'OK', it carries out check of command code. If the three-character command code is valid, function FINDUNIT is invoked to search and obtain the pointer to the particular unit on which the transaction is to be carried out. If the unit is found, then the main module invokes one of the following procedures depending on command :

(i) I – INSERTUNIT.

(ii) D – DELETEUNIT.

(iii) UI – UPDATEINVENTORY.

(iv) UA – UPDATEACTION.

(v) Q -- INTERPRETQUERY.

The transactions are carried out one by one till a predetermined number of transactions or till EOF condition is reached in the TRANS file.

6.1.2 Input

All transactions are sequentially read from the queue. The main module reads the following portion of the transaction :

Data-item	Type	Width	Example	Remarks
Commandcode	Char	3	*2ƀ	
Unit-id	Integer	3	212	
Unit-name	Char	9	INFƀREGTƀ	
Field-separator	Char			

6.1.3 Output

This module prints out the following in case of successful transactions :

(i) Transaction number and type of transaction, and

(ii) Total OVH factor.

(iii) It prints out appropriate error message in case the transaction contains incorrect command codes or if the keyunit on whom the transaction is to be carried out is not found in the structure of the database.

6.1.4 Algorithm

```
Start
Read Pasdirectory, Actdirectory, Subdirectory, Inventdirectory
Create a new pointer to the root of the tree initialise all links to Nil
Set Transaction count to 0
```

```
While Transaction count = Maximum Transaction or till EOF DO
Set Count to count + 1
Call procedure password
If passwords is OK
Then Read command code and keyid from Transfile
   Check validity of command code
   If command code valid
   Then call function Findunit (root, keyid')
          Set keyunit to Findunit
   If keyunit is not Nil
       Then Read Keyname
       Depending on Command code call
       appropriate procedure for processing
   else print keyunit not found
   else print error message of command code
   else end transaction
end (of while loop)
stop
end of algorithm.
```

6.2 Procedure PASSWORD

This module is invoked by the main program. Before processing any transaction this module checks the validity of branch code and related passwords. A directory containing valid passwords for different branches in a HQ which is held on a disc file to read in by main program. The directory can be modified as and when required.

6.3 Function FINDUNIT

Function FINDUNIT searches through the tree structure recursively to locate the node (unit) to which the transaction pertains. It is invoked by the main program to which it returns a pointer to the particular node which has got the unit ID that matches with the specified key.

Procedure INSERTUNIT

Procedure INSERTUNIT is invoked by the main module after it obtains the pointer to the predecessor unit. The procedure reads the successor unit's identification and name. It then inserts the successor unit to an empty link of the given predecessor. If no link of predecessor unit is empty, it gives message that the predecessor is full of links.

6.5 Procedure DELETEUNIT

Procedure DELETEUNIT finds out the link between the identified predecessor and given successor and deletes the successor if the successor exists under the predecessor and makes the corresponding pointer to NIL. This procedure is invoked by the main program on identification of the corresponding command code.

6.6 Procedure UPDATEINVENTORY

This module is invoked by the main program if the command is to update inventory. It reads, validates and updates the items of inventory record one by one. As such only valid item name and corresponding quantity are processed, while errors are flagged and indicated if the inventory item is incorrect. This module also checks the inventory level after updating and gives an indication of the items if the level has fallen to 80 per cent or below their authorised levels.

6.7 Procedure UPDATEACTION

This module is invoked by the main program. It validates the data and only proven data is used for updating. The salient functions of the module are :

(i) Validation of all data fields (except text) and printing error message if applicable.

(ii) Repacking date and month fields and storing for use of other modules.

(iii) If the transaction is valid, looking for an empty location in the ACTREC array and appending the transaction.

(iv) If ACTREC array for a particular unit is full displaying a message.

6.8 Procedure INTERPRETQUERY

This procedure is the 'Central Query Handler'. It interprets the queries on database, and depending on the type of query, it invokes one of the specific procedure to service the query. The procedure INTERPRETQUERY itself is invoked by the main module on identification of a transaction as a query.

6.9 Procedure ANSWERINVENTORY-QUERY

This procedure is invoked by the query interpreter to answer the queries related to the entire inventory or specific item(s) of inventory held by the unit in question.

6.10 Procedure ANSWERACTIONQUERY

Procedure ANSWERACTIONQUERY answers queries related to actions of the unit in question. Depending on values of query code it provides all actions pertaining to the unit or only actions during the specified time interval. It is invoked by the procedure INTERPRETQUERY with 'From Date & Time' and 'To Date & Time' parameters. These parameters are zeroes when all actions are required and have specific values when actions during specified time interval are required.

6.11 Procedure GIVESTRUCTURE

Given a key for a particular formation/unit, this procedure scans and prints the structure under this unit (node) recursively down to the regiment level and gives the structure under the specified unit.

6.12 Procedure GIVEORBAT

This procedure is called by the central query interpreter to obtain the immediate successors of any given unit (node) in the structure (the word 'ORBAT' implies units/ sub-formations under direct command).

7. SYSTEM RESPONSE

A sample of 148 transactions in random order has been processed to study the system response time. A sample of transactions has been shown at Appendix C . The time taken for each of the transaction mentioned above has been calculated based on a pseudo factor OVH. Each statement within the programme has been allocated a processing time of one OVH unit. Thus depending on the number of times an instruction is executed either interactively or recursively, the overhead is incremented in the pseudo master clock of the programme. The response time characteristics for each transaction has been printed in the output as shown at Appendix 'D'. However, this OVH for data can be stored in a file for statistical analysis. The transactions have been divided into the following basic functional groups as shown in Table 1.

The response time curve based on overheads and number of transactions for each type of transaction along with their respective average response time and standard deviation can be plotted for any required analysis.

Table 1. Basic data for each type of transaction

Sl. No.	Type of transaction	No. of transactions implemented	Minimum time of service	Maximum time of service	Average
1.	INSERT	53	55	683	313.85
2.	DELETE	6	90	485	216.82
3.	UPDATE INVENTORY	44	72	780	430.25
4.	UPDATE ACTION	10	89	737	355.30
5.	QUERY INVENTORY	16	71	702	410.25
6.	QUERY ACTION	10	75	727	344.25
7.	QUERY ORBAT	9	45	556	274.33
8.	QUERY STRUCTURE	1	—	677	—
		Total	verage of	ieans	335.8

8. CONCLUSION

Keeping in view the user requirement as well as technical parameters, a database model using PASCAL has been developed. Since the 'Tree' structure is generated/scanned using recursive techniques and 'Pointers' facility of PASCAL, any size of 'Tree' can be organised/accessed by the user. It is visualised that in distributed environment where transactions will be processed at distant location, PASCAL or 'C' language will be a most suitable language both for writing communication control software as well as organising database structure.

The system also caters for manual interface where assessment of operational situation is carried out by competent staff and sent as free text in 'Activity' transactions. Critical situations can be flagged by special indicator on the recipient's terminal so that concerned staff can take timely and appropriate action. The concept projected in the paper can be extended by suitable modifications to use in Navy, Air Force, R & D or Civil Organisations. It is appreciated that development of an actual database model to meet operational requirements of the Army may take 20–25 man years.

ACKNOWLEDGEMENTS

This project was the combined effort of a team of five officers from Defence and one scientist from DRDO while undergoing a course of real-time design at TIFR, Bombay in August 1981.

BIBLIOGRAPHY

Shave, M.J.R., Data Structure, (McGraw Hill, New York), 1975.

2. Date, C.J., An Introduction to Data-Base Systems, (Addisson-Wesley, Reading), 1980.
3. Martin, J., Principles of Data-Base Management, (Prentice-Hall, Englewood Cliffs), 1977.
4. Wirth, N., Algorithms + Data Structure = Programs, (Prentice-Hall, Englewood Cliffs), 1976.
5. Joglekar, V.K., Query Language, Term Paper, (Univ. of Nebraska, Lincoln), 1977.
6. Schneider, G.M., *et. al.*, An Introduction to Programming and Problem Solving with PASCAL, (John Wiley, New York), 1978.

APPENDIX A

Sample List : Activity Code

Code	Description
ADV	ADVANCE/ADVANCING
ATT	ATTACK/ATTACKING
BLD	BUILD/BUILDING
BRC	BREACHING
CONC	CONCENTRATING
CAP	CAPTURED/CAPTURING
DES	DESTROYED
DUM	DUMPING/DUMPED

APPENDIX B

Sample List : Activity Subject Code

Code	Description
ACT	AIRCRAFT
AAD	ANTI-AIRCRAFT DEFENCE
ATK	ANTI-TANK
BDE	BRIGADE
BTY	BATTERY
MIS	MISSILE
TRK	TRACK
LOG	LOGISTICS

APPENDIX C

Sample Transactions

L-GAIN *UI 248 UNIT	GUNS + 45 RSET + 56 PERS + 790 $
L-GAIN *UI 237 UNIT	TANK + 34 RSET + 25 VEHS + 40 PERS + 770 $
L-GAIN *UI 402 UNIT	GUNS + 24 BEHS + 56 RSET + 42 PERS + 810 $
I-RAJA *UA 226 UNIT	03 30 04 30 CONTPS 860A-LIKELY BUILD-UP $
I-RAJA *UA 237 UNIT	09 08 10 30 DUMAMN 002B-IEK RI AP MOV FDW $
I-RAJA *UA 552 UNIT	09 15 16 45 MOV TNK 0800-NW OF TIBBA DECEPT $
I-RAJA *UA 424 UNIT	10 04 18 30 DESGUN 005D -LOST IN AIR RAID $
I-RAJA *UA 543 UNIT	10 06 19 30 JAMVHF 000E-HY DAMMING COMDNET $
I-RAJA *UA 402 UNIT	10 12 18 45 BLDBRG 001A-BRG NEAR COMPLETED $
I-RAJA *UA 282 UNIT	10 15 04 30 DESBRG 002D-A CORPS TAPTI/BET WA $
I-GAIN *UI 511 UNIT	TANK-05 RSET-10 $
Q-OPEN *01 1222 UNIT	GIVE HOLDINGS FULL $
G-WAIT *QA 222 UNIT	GIVE ACTIVITY FULL $
G-WAIT *Q 017 MIDDLE HQ	GIVE ORBAT $
G-WAIT *D 112 LOWER HQ	212 UNIT $

APPENDIX D

```
                          TRANSACTION NO 56
                         UPDATE OF INVENTORY

214 UNIT
      GUNS UPDATED TO 12
      VEHS UPDATED TO 45*
      ERROR IN ITEM CODE RSET THIS FIELD SKIPPED
      PERS UPDATED TO 720*
      (OVH FACTOR : 90)
-----------------------------------------------------------

                          TRANSACTION NO 68
                            QUERY-ORBAT

SUCCESSORS OF 42 MIDDLE HQ
   1.   112 LOWER HQ
   2.   119 LOWER HQ
   3.   101 LOWER HQ
   4.   117 LOWER HQ
        (OVH FACTOR : 58)
-------------------------------------------

                          TRANSACTION NO 85
                           UNIT - DELETION
125 LOWER HQ NOT FOUND AND NOT DELETED
        (OVH FACTOR : 291)
   ----------------------------------------

                          TRANSACTION NO 114
                            QUERY-ACTION
TOTAL LIST OF ACTION OF 212 UNIT
DATE        TIME        ACTIVITY    QTY            TEXT

0828        1305        BOM ACT     6          F-16 AC-NW TWO SORTIE
0828        1305        BCM ACT                C-PT 1 BRG-2 BLON S
0           0           0
0           0           0
                           (OVH FACT OR : 79)
```