# DEVELOPMENT OF AUTOMATED LAB MANAGEMENT SYSTEM ON ORACLE

Dissertation submitted to Jawaharlal Nehru University
in partial fulfilment of the requirements
for the award of the Degree of

**MASTER OF TECHNOLOGY**



**MARTHALA VASAVI**


**SCHOOL OF COMPUTER & SYSTEMS SCIENCES**
**JAWAHARLAL NEHRU UNIVERSITY**
**NEW DELHI-110067**

**1989**

**INDEX**

**CERTIFICATE**

This project titled" ***AUTOMATED LAB MANAGEMENT SYSTEM*** "has been carried out by me under the supervision of Mrs.Anjali Raina ,Research Engineer ,C-DOT and Lt Col.C.P.C.Nath, Associate Professor , SC & SS , Jawaharlal Nehru University, New Delhi .

This work submitted in partial fulfillment of the requirement of the MASTER OF TECHNOLOGY degree of Jawaharlal Nehru University ,New Delhi is original and has not been submitted in part or full for any degree or diploma in any other institution .

M.Vasavi.
( M.VASAVI )

Candidate

Anjali Raina

Mrs.Anjali Raina ,
External Supervisor ,
Research Engineer ,

C-DOT ,New Delhi .

C.P.C.Nath

Lt Col.C.P.C.Nath ,
Associate Professor ,
School of Computer &
System Sciences ,
Jawaharlal Nehru Univ.,
New Delhi .

Prof.P.Mukherjee , Dean ,
School of Computer & system Sciences ,
Jawaharlal Nehru University ,
New Delhi.

**ACKNOWLEDGMENTS**

It is pleasure to acknowledge Mrs. Anjali Raina, Research Engineer, SQA group, C-DOT, for giving me full freedom, necessary guidance and encouragement to work on my project at C-DOT.

I am also thankful to Mrs.Kanvinde,Group Leader , SQA, Mr.Shukla ,Research Engineer ,C-DOT and Mr.Srivatsava Research Engineer , C-DOT for giving constant assistance, cooperation and for the interest shown on my project.

I gratefully acknowledge the valuable advises, suggestions, inspiration and motivation rendered by Prof.C.P.C.Nath during the work on my project.

Finally I am delighted to acknowledge my debt to all those who are directly or indirectly involved in giving this final stage to my work and report.

[ M.VASAVI]

## PREFACE

The title of the thesis is DEVELOPMENT OF AUTOMATED LAB MANAGEMENT SYSTEM ON ORACLE. The aim is to develop an application program viz AUTOMATION OF THE LAB MANAGEMENT SYSTEM at C-DOT,Center for Development Of Telematics, in an RDBMS environment.

This thesis is organized at three distinct levels.

It does not suffice just to develop the application program on any RDBMS, it was felt that a comparative study has to done among a selected few RDBMSs before the development of the application program on an optimal RDBMS is carried on.At the time of doing this project ORACLE and INGRES are two of the few RDBMSs having optimal pedigrees of the business.

Chapter 2 is dedicated for this purpose. It compares various technical features of ORACLE and INGRES and the justification is given for choosing ORACLE as the working environment.

Merely choosing the right RDBMS one cannot start developing the application program unless he/she is familiar with that RDBMS. So the second part, chapter 3 discusses in a brief yet clear way about ORACLE. It describes the ORACLE technology, the productivity tools provided by it and the error handling techniques in brief.

The last part of this report describes the development of the application program, viz AUTOMATED LAB MANAGEMENT SYSTEM. It discusses how the program is first developed using data flow diagrams (DFDs) and then how each DFD is implemented on ORACLE.

Appendix A contains a list of relations designed for the implementation of ALMS.

Appendix B has the screen printouts of various SQL*Forms that are designed.

Appendix C gives a list of comparative study between ORACLE and INGRES

**ABSTRACT**

Labs are provided in many software organizations to aid the engineers to test the software that was developed by him/her. So automation of lab management is essential for the efficient utilization of labs. Automation of lab management is done on ORACLE RDBMS using various tools provided by the system.

ORACLE is prefered to INGRES because of its special features which include a larger record and fields in ORACLE, no page level locks, better usage of indexes and a better OnLine Transaction Processing.

# 1. INTRODUCTION

C-DOT is a S/W organization for the development of telematics. In all such S/W organizations testing of the software developed by the engineer is important before it could be released for the application. For this purpose of testing, labs are provided to give a real time environment to test the packages. However, there is always a rush of engineers wanting to test their models in the labs.In order to manage such requests from the engineers to use the labs, the LAB MANAGEMENT has come up.

Earlier, the lab management was done manually. Lab managers allot shifts to engineers on a time slot basis and the scheduling was done on the basis of the priority of the job. There was always a overhead on the lab manager to keep track of all the data regarding labs such as requests, allocation of the shifts, log out data, existing patch links and etc.

In order to avoid the short comings faced by the manually operated lab management, it was decided to automate the lab management system. The data regarding the lab management is stored in the tables, and can be accessed through the various tools provided by the ORACLE such as SQL*Plus, SQL*Forms, SQL*Reports and etc. In this part of the book we shall consider in detail the need for the automation and how it is achieved.

But before this, a comparative study was done between INGRES and ORACLE to choose the appropriate RDBMS to develop the package. Conclusion was drawn that due to many distinct features that ORACLE has and which INGRES does not, ORACLE is preferred.

## 2. COMPARATIVE STUDY OF ORACLE AND INGRES

Before the development of any application package, a right RDBMS which could match the needs of the application has to be chosen.

Two of the efficient RDBMS at present are ORACLE and INGRES. Performance level of these two RDBMSs are almost at the same level except for minor differences. In some aspects ORACLE outperforms INGRES and in some cases INGRES gives a better performance. Here we shall consider in brief, the performance characteristics of ORACLE and INGRES.

Firstly, ORACLE has the rolling forward facility which the INGRES RDBMS does not have. That is, when, for example, system failure has occurred, ORACLE's system monitor after rolling forward all the committed and uncommitted transactions, rolls backward the uncommitted transactions, the information about which is stored in the rollback segments.

ORACLE makes use of indexes (if one is created) very efficiently, while performing the queries. It outperforms the AI search techniques, which is used in INGRES to perform a query, provided index is created on the fields in the WHERE clause of the query. But the more the number of indexes per table, the more is the problem to maintain the table.

A record in INGRES RDBMS can accommodate fewer fields than that in ORACLE. The same applies even to the field lengths. In INGRES each record have a maximum of 127 fields

per record and 2000 character per record where as in ORACLE there can be 255 fields per record and 128K characters per record.

In INGRES page level locks are the lowest level that the system can provide. If a user wants to access a data from a page, firstly the system locks the page thus disabling any other user to access different rows in the same page. This is not the case in case of ORACLE, as the system locks only the rows that the user is accessing, and not the whole page.

ORACLE provides shared update locks. It does not obstruct users from reading the same row in the same page. INGRES places locks even if the user wants to read the record. So the user has to wait till the first user finishes reading the record. This is not the case in ORACLE.

Occurrence of deadlocks is more frequent in INGRES than in ORACLE as INGRES provides even read locks. That is, in INGRES a read lock is placed on every page that the user wants to read. Consider the following situation:

Suppose user A locks the page 1 to read certain record of that page at time t1 and user B locks page 2 to read few records of that page at time t2 ($t1<t2$). Now at time t3, ($t3 > t2$), user A wants to access records of page 2 and user B that of page 1, a deadlock would result. THis is detected by INGRES, and it kills the transactions of one user. Such situation will not arise in ORACLE as there are no read locks

which enables many users to read the same rows at the same time.

CONCLUSION : From the comparative study between ORACLE and INGRES, the conclusion was drawn that ORACLE is better suited to develop the package. In appendix C, a table of comparative study between INGRES and ORACLE is given.

In the next section a overview on ORACLE RDBMS is considered.

# 3. ORACLE RDBMS

In this section, a brief overview of the ORACLE RDBMS is given. It discusses features of ORACLE technology, various productivity tools offered by the system and the like.

## 3.i. ORACLE TECHNOLOGY

The two major aims of relational database (rdb) are :

1. Need for the OnLine transaction processing providing a high performance and fault tolerable with relational productivity.
2. Enterprise wide computing - instant access to information on mainframes, minicomputers, microcomputers.

ORACLE delivers the distributed OLTP, efficient decision support and all the benefits that are inherent to a SQL relational database. We shall see how ORACLE works to achieve the two major aims, mentioned above, of the rdb environment in the subsequent pages.

HOW OLTP IS ACHIEVED BY THE ORACLE DATABASE SYSTEM ?

Relational database environment has the following benefits:

1. Non-procedural access - Making the system more user friendly as the user need not bother about how the results are retrieved.

2. Standard access - SQL , a simple, comprehensive language that is useful for all database activity.

3. Reduced maintenance cost - Reprogramming not required incase of any physical changes to the data storage.

4. Highly flexible - The database system adjusts immediately to changes in the business.

5. Reduced application backlog - application development is faster and less time is spent maintaining code.

6. High transaction volumes - often hundreds of transactions per second.

7. Fast response rates - Usually a subsecond.

8. Fault tolerance - virtually no down time .

9. Large, online user communities - often many hundreds of users.

10. Very large databases - sometimes hundreds of gigabytes in size.

Most of the DBMSs provide what the low-end decision support needs viz the first five benefits. ORACLE is suitable to satisfy the high-end systems as it is capable of handling

all the ten benefits that the relational environment can provide to the users.

ORACLE enjoys the joint advantages of both the relational as well as traditional DBMSs. ORACLE provides high performance that a traditional DBMS provides along with the high productivity that is inherent to the relational environment.

In the next section we shall see how the ORACLE achieved the demands of high productivity as well as high performance.

## OLTP PERFORMANCE

### ARCHITECTURE:

ORACLE delivers high performance through a combination of optimal client / multiserver architecture and concurrency control mechanism that uniquely supports simultaneous online transaction processing and decision support.

Client/server architecture provides the following advantages :

1. Cost effectiveness
2. Expandability
3. Remote access
4. Flexibility

ORACLE was the first commercial rdb to offer a full client/server architecture. This architecture of ORACLE precisely divides user code (client) and database code

(server). Client handles all user interface whereas server manages all the database actions. With this architecture one can run an application on one machine while simultaneously running the database on the other.

Incase of multiserver architecture, the system gives a very good response rates and the transaction volumes are high as the hardware of such systems uses several loosely coupled or tightly coupled CPUs thus avoiding the performance constraint by a single CPU.

ORACLE uses multiserver architecture to deliver a high performance. Logically a server is dedicated to each user though physically all the clients share the same server code. This technique of sharing the multi-threaded re-entrant code saves memory.

Now users can do work simultaneously on symmetric multiprocessor computers. Performance is directly related to the number of CPUs per computer and the horsepower of each CPU.

In a single server architecture the system cannot tap more than one CPU's powerat a time. Though the single server performs all the database work one can execute on only one CPU at a time. Thus with one server the performance is limited to the speed of one CPU though the computer has multiple CPUs.

Moreover ORACLE's multiserver architecture delivers complete scalability. This architecture provides incremental throughput as one adds CPUs.

I/O OPTIMIZATION:

The requirements of OLTP performance donot create any bottlenecks during I/O transactions.These I/O bottlenecks are taken care of by ORACLE's efficient I/O algorithms. Here we shall discuss how these bottle necks are avoided by tracing the actions of a transaction.

ORACLE processing of READS:

It is the user who actually performs all the database operations including read because it is he/she who executes both the server code and client code. While retrieving the blocks that contains the required rows the first point to be considered is how fast it can be done. For this purpose ORACLE along with the standard access paths uses the following access methods :

1. Multiple indexes.
2. Data stored in index.
3. Clustered table.

ORACLE query optimizer decides as to which access method is best to retrieve the required blocks.

Once the required blocks are identified, these blocks are stored in the shared memory cache after the user reads the necessary blocks from the database. Subsequent request for the same blocks by the same user or other user need no more I/O operations and the retrieval this time is done from the memory.To boost the performance ORACLE can be configured to use any size memory cache that one's application asks for.

In ORACLE multiple block reads can be done in one I/O request. Thus a query that requires 1000 blocks could perform 10 I/Os of 100 blocks each rather than performing 1000 I/Os of one block each. This characteristic of multiple block access in a single I/O operation boosts the performance of the system as low number of I/O requests are issued.

ORACLE processing Writes :

The transactions are committed only after the required blocks are brought into the memory.

ORACLE uses Redo log files to guarantee that the committed transactions remain permanent. A process, redo log writer (LGWR), protects the changes made to the blocks in the memory by recording these changes in the redolog files.

Incase of system failure, ORACLE automatically recovers the database upon startup using the online redolog files. The entire media failure protection can be achieved by a process called Archiver process which copies all the redo log files to the tape. Thus the archiver logs has a complete history of all the changes that has been made to the db since the last backup.

The LGWR process is very efficient because it sequentially writes only a minimal information in the redolog file about each transaction. Moreover sequential write to the redolog file is much faster than the random write to the database file. This sequential write of a transaction's data

is done in a single I/O operation. Hence the performance is much greater than most other DBMSs which requires more than 1 I/O per transaction.

Transactions are piggybacked when multiple transaction request to commit together. This makes on average less than one I/O per transaction boosting the performance of the system. Rather than serving each transaction one at a time LGWR writes all the transactions wishing to commit simultaneously.

A process viz DBWR, ,database writer, writes the modified blocks from memory to the database. Each time a block is modified in memory, DBWR does not write the modified block from the memory to the database immediately. This writing is deferred until the memory cache cannot hold any more requested blocks. The DBWR writes the modified blocks from memory into the database on the least recently used basis thus boosting the performance by ensuring that the recently modified blocks remain in the memory.

To summarize, ORACLE's I/O algorithm guarantees :

1. Minimal data is written very quickly.
2. Maximum of one sequential write is required per transaction.
3. Frequently, less than one sequential write is required per transaction.
4. Commits donot require that changes be written to the database.

DATABASE I/O OPTIMIZATION : It can be achieved at the global database level by assigning the database objects like tables, indexes, temporary segments (db space used for sorting and ordering), rollback segments (db space used for transaction recovery) and redolog files to separate devices to further increase I/O concurrency. Moreover all the objects can span multiple devices.

NETWORK OPTIMIZATION:

In the distributed environment, communication channel bandwidth imposes restriction on the transaction processing between the client and the database server. This overload is removed in ORACLE by its unique array interface and PL/SQL procedural transaction processing language.

While other DBMSs move data between application programs and database one row at a time, ORACLE array interface allows this transaction of multiple rows by transferring data in batches instead of row by row.

PL/SQL also reduce the demands on narrow communication channel by grouping multiple DBMS requests into a single request. Before returning the control to the user, ORACLE executes the PL/SQL procedure hiding all the intermediate results from the user.

ROW LEVEL CONCURRENCY CONTROL:

ORACLE delivers high performance by maximizing data access by multiple and concurrent users without jeopardizing data integrity. This is achieved through the following 3 mechanisms :

1. Row level multiversion read consistency.
2. Row level locking.
3. Sequence number generator.

Row level multiversion read consistency :

ORACLE's multiversion snapshot model allows queries to read without locks and consequently queries donot block both queries and update and also updates donot block queries.

When ORACLE updates a row, it also records enough information to generate a pre-update snapshot of the row in memory or in rollback segment areas of the database. Rollback segments are used for read consistency to ensure that a query uses a consistent image of the db as the start of the query.

By executing queries without locks, one can update a row while other can read the same row at the same time from the snapshot i.e. queries donot block updates thus achieving high concurrency which means no waiting which inturn means high performance.

Unlike in many other DBMSs, in ORACLE there is no tradeoff between consistency and concurrency.In many other RDBMs one is achieved at the cost of the other.

For example, to process a join same rows of one table may be read more than once. Thus the row's value should be protected rom update during the process of join. To achieve this, in many DBMSs, locks are imposed on all the blocks read by the query during the duration of the transaction. Though this ensures read consistency concurrency is reduced as no one can update those rows until the join is resolved.

Conversely, to achieve high concurrency, other DBMSs turnsoff the readlocks but this would result in the wrong rows being returned during a join operation.

These problems are removed in ORACLE by its multiversion snapshot model of read consistency that gives high concurrency and high data integrity.

Stable query response time is yet another advantage of ORACLE's read consistency algorithm. Since quries never wait for update transactions or other queries, there is little variation in response times as the number of current updates increase.

Inmost DBMSs deadlocks can occur if two or more transactions update two different rows and then each attempts to read the row that the other has locked. This is not the case in ORACLE as it doesnot place any read locks.

level locking :

Locking in ORACLE is done at row level with the following unique advantages :

1. Users can update rows in the same page .
2. Users can place unlimited number of locks .
3. Users avoid lock escalation deadlock :
4. Performance is sustained even with the data and index level hotspots :

(Hotspots, the concentrated portions of the table, occurs when multiple users update rows of the same table or if data distribution happens to place the rows to be updated in the same physical space.)

5. Inserts are multi-threaded.

NO - WAIT SEQUENCE NUMBER GENERATORS :

ORACLE generates unique numbers for all forms of primary keys without waiting or programming and it does so without locking.

With programming, the typical method of generating a sequence number, say order_no, is lock order_no table, increment the current order number and release the table. Each user has to wait his turn to lock the table which is a single threaded operation.

HOW ORACLE AVOIDS ROW LOCK OVERHEAD :

ORACLE elevates the concept of row lock waits to the transaction level. For example, user A has locked ten rows of

a table and user B wants to update five of these ten rows. ORACLE does not keep track of the five locks that has to be released in order to proceed with B's request, instead ORACLE just remembers that user B is waiting for user A's transaction to finish. Consequently ORACLE manages one transaction instead of five separate row locks. This provides the usage of unlimited number of row locks and also no row lock escalation which makes ORACLE to deliver a highest possible concurrency even under heavy loads.

FAULT TOLERANCE:

OLTP applications besides processing fast must also be available at any time. With ORACLE need not be shutdown and H/W failures that shutdown the system can be prevented.

Online database administration minimizes the down time:

With ORACLE one can perform online database configuration, diagnosis, backup and recovery without interrupting the work of the database users running any application programs.

- Online database configuration :

One can tune performance in many ways including addition of tables and columns of tables, alteration of existing columns, indexes, clusters. The feature of ORACLE to manage the database space online permits one to move files from disk to disk or move infrequently used portion of database to tape or even expand the database by adding the

files online. All these changes are transparent to users and applications.

o Online diagnosis :

ORACLE's online performance monitor determines the status of the database at any point in time. The data can be used to improve the transaction processing for maximum performance. The data that the online monitor displays :

1. I/Os by user an by files.
2. Locks at the user and the database level.
3. Rollback segments information.
4. System statics such as logons, cursor opens, database calls, and buffer usage.
5. User session information.

o Online backup:

Complete or partial online backup to the database is allowed without impeding the work of any Oracle user, even those updating the database.

Without degrading the system performance, the logging of all changes to the redo log files provides space efficient protection against all forms of system failures.

Online backup is fast as it is done by the O/S. ORACLE's online backup can take consistent backup and does not impact concurrent OLTP operations.

If the database needs to be recovered, a consistent snapshot of the database can be created from the tape copy of

the database plus the redo logs that are archived during the back up.

o Online recovery :

While portions of the data are online and accessible, one can recover the failed section of the same database by taking the failed section offline, recovering that section and bringing back the section online preventing the database shutdown.

After CPU or media failure, database recovery is the process of rolling back the uncommitted transactions and rolling forward the committed transactions. As a first step of recovery a separate database process, the system monitor (SMON), uses redolog information to rollforward all committed and uncommitted changes. As a second step, the SMON identifies the uncommitted changes using information in the rollback segments.

Recovery time is controllable. Recovery time depends on the amount of redolog information that must be applied to the database. Data in the redolog files is no more needed once the modified blocks have been written from the memory to the database. A check point is the act of flushing all modified blocks from memory to the database. As one can control how much redolog information is generated between check points, the time for recovery of the database after CPU failure is controllable.

For disk failures, recovery time is directly dependent on the amount of redo log information generated since the last backup. as the frequency of the backup is controllable and also since the database backup has no impact on performance, the time it takes to recover the database after disk failure is also controllable.

H/W Fault-tolerance maximizes uptime :

The fault tolerant capabilities of ORACLE maximizes the availability of the database information despite H/W failures.

Protection against user and program failures: With ORACLE the user failures such as intentional user aborts and the program failures such as stalled or hung processes occurrences automatically recovers the database without shutting down or corrupting the database.

Protection against disk failures : If a database disk should fail the failed disk can be recovered online, restoring it to its previous consistent state without interrupting applications accessing other disks.

Protection against CPU failures : The shared-disk configuration allows any number of computers to run ORACLE client/multi-server DBMS. If one of the computers fail, with ORACLE one of the remaining computers automatically recovers all transactions of the failed computer. At the same time one can move the users from the failed computer on to any of the functioning computers.

Protection against network and node failure : As each node operates independently, the network failures such as node failures, line crashes or transmission errors donot force to shut down other nodes of the distributed database (ddb) and also they donot corrupt the database of the failed node.

Central node distributed have a single point of failure - the global dictionary located at the central node that cuts off access to all data on the network if the central node fails. With ORACLE the dictionary is distributed among nodes in the network so that the failure of any one node doesnot affect the accessibility of data located on other nodes.

Fault Tolerant H/W

In addition to ORACLE's complete set of online and fault tolerant features, ORACLe runs on a number of fully fault tolerant computers and disk monitoring systems to provide as much fault tolerant as one requires.

VERY LARGE DATABASE SUPPORT

ORACLE supports very large database common to OLTP environments through a combination of no limits and useful database management utilities.

o Virtually no limits :

The size of the database is only limited by available storage. An ORACLE database or a table can span as many physical disk devices as one make available. There are no

artificial limits on the number of valid or active users of a database and also row locks at the user, table or database level may extend to any limit. Moreover there is no restrictions as to the number of simultaneous transactions per database.

With ORACLE, database can exceed the amount of available storage - virtual storage capability. One can take a portion of database online when required and take it offline when not required. When the sections are offline these sections can stored on magnetic tape.

o Database management ulilities

The following three major features are provided by the ORACLE to control users and space :

1. One can control access or types of access to tables down to field levels by a combination of views and GRANT commands.One can also audit access to the database and to the tables. One can also determine when the given user has logged on and off, what operations are performed against what tables and whether the operation succeeded.

2. One has direct control over who uses what databases and how much database space. Space allocations are dynamic and many defaults can be set.

One can also control the space within a table - whether the table grows in small or large increments, how much space is reserved in each block of row expansion and what empty space within a block is reused.

3. The dynamic ORACLE data dictionary, where all the usage information is available, is tightly integrated with the operation of the database. Any change to the database is immediately reflected in the data dictionary. From the active data dictionary, a range of information is available including what users have access to the database, what tables or other database objects have been created, how much space has been consumed and what privileges have been granted.

OLTP SUMMARY : ORACLE extends the relational productivity benefits of greater flexibility, ease of use and non-procedularity in accessing database data, to the online transaction processing arena with a combination of high performance, seamless fault tolerance and very large database support.

## THREE TIER, HETEROGENEOUS, ENTERPRISE-WIDE COMPUTING

Despite the different tiers of computing platforms (at corporate level on mainframes, at department level on minicomputers and at workshop level on microcomputers/work stations) an despite the heterogeneity in the computer platforms (mainframes, minicomputers, microcomputers, work stations), all areas of organization need to access and to share information throughout the enterprise inorder to take full advantage of their information resources.

1. Portable solution

ORACLE RDBMS is one stop solution to three tier, heterogeneous, enterprise-wide computing. Oracle runs on all major computer platforms.

Advantages gained with a portable RDBMS :

1. Increased flexibility.
2. Reduced development cost.
3. Reduced training costs.
4. Increased H/W independence.

2.Distributed database solution

ORACLE's heterogeneous , ddb solution provides the last ingredient for enterprise wide computing - data sharing. Coupled with ORACLE's SQL*Net and SQL*Connect, all ORACLE database and many non-ORACLE database can share information through the organization.

The benefits of the distributed database include :

1. Flexible departmental computing : Ddb systems store data where it is most likely to be needed - reducing network traffic and maximizing data availability in case of network failure - while permitting authorized users throughout the organization access to the same data.

2. Simplified application programming : Application can access data stored on multiple computers with same ease as if the information was stored on the same computer.

3. Simplified data sharing : Ddb capabilities unify dissimilar applications on separate computers by enabling them to operate on a common logical database.

Since ORACLE runs on and connects between a large variety of platforms, application can access information from different environments as easily as from homogenous environment.

3. Distributed architecture

The three characteristics of the ORACLE's distributed architecture are :

o Location transparency : All data seems to reside on the local database. User need not specify the physical location of the data. ORACLE's data dictionary performs this task and retrieves the data the query refers to from the table name specified by the user. The retrieved data may be a remote

data or local data or a combination of both. One can move the data among the nodes without recoding the query.

o Site autonomy : A network is said to have site autonomy if there is no central process or node responsible for control over system-wide functions such as routing, scheduling, query optimization or deadlock detection.

Advantages of site autonomy include :

1. Better, local control over data definition and security.
2. Fewer interdepartmental dependencies.
3. No central points of failure.
4. Easier failure recovery since each site can be recovered independently.
5. Easier system growth.

o Network independence : ORACLE's ddb supports a number of LANs and WANs. It can even operate over several be replaced without recoding the applications.

ORACLE makes enterprise-wide computing a reality by providing ORACLE on all types and sizes of computers and connecting these heterogenous systems with a sound ddb architecture.

SUMMARY : OLTP and enterprise-wide computing are the power targets in the relational database industry. ORACLE RDBMS with transaction processing option is one DBMS that addresses both. ORACLE has the performance, fault tolerance and large database support required to deliver high transaction volume and fast response rates for hundreds of users. Its portability, distributed database architecture and breadth of network of network protocol offerings make it the ideal solution for three tier heterogeneous enterprise-wide computing.

## ii. PRODUCTIVITY TOOLS FOR APPLICATION DESIGNERS AND DEVELOPMENT

Oracle provides developers with a complete environment for designing and implementing robust production applications. Using Oracle' application development tools, one can create sophisticated transaction processing, reporting, menuing systems, all without programming.

SQL*Forms :

Complete application without programming : Forms-based transaction processing applications can be done quickly and efficiently, all without programming.

Effective prototyping : SQL*Froms' unique non-procedural approach promotes effective application protyping thus letting one to refine his/her application as one builds.

Open Architecture: Application can be built as per the individuals' interest without any restrictions. Database tables can be accessed by simple SQL statements.Embedded procedural macros or routines written in COBOL, C, FORTRAN, PASCAL etc are allowed to customize any aspect of application functionality.

Development flexibility: Development backlog can be eliminated and applications can be kept as current as the information needs since it takes very less time to adjust the

SQL*Plus

It delivers a full implementation of SQL as well as powerful report-writing and data transfer capabilities. SQL statements and formatting commands can be executed interactively or from stored command files.

SQL*Report

It is very useful to create everything from basic text to sophisticated multi-query reports.

SQL *Design Dictionary

It is a Computer -Aided System Engineering (CASE) system built on the ORACLE RDBMS in order to manage the application development process. It, besides documenting every component of the application developed, also performs consistency and quality checks throughout the analysis and design process.

Programmatic Interfaces

ORACLE supports two types of programmatic interfaces- precompilers and procedural interfaces. Precompilers lets one to access and manipulate data using familiar programming languages in the form of embedded SQL statements. Precompilers convert such embedded SQL statements into the appropriate programming language source code.

Alternatively, SQL statements can be executed by the ORACLE call interface to pass SQL procedure calls to BMS.

Besides these tools there are many other tools ORACLE provides. Few of them are :

SQL*Calc, to make portable and consistent faces across machines;

SQL*QMX, which combines dynamic query facilities an easy yet powerful report writer;

Spreadsheet Interface which delivers the tages of ORACLE RDBMS coupled with the capabilities of an to learn spreadsheet

SQL*GRAPH that provides graphics facility .

## iii. ERROR HANDLING TECHNIQUES IN ORACLE RDBMS

Errors that might be encountered while using ORACLE corporation programs are broadly divided into three categories :

1. ORACLE errors : These errors are detected by ORACLE RDBMS and might occur while running any ORACLE program. Each Oracle error message has a prefix ORA.

2. Product-specific errors : These errors are specific to on product.

3. System-specific errors : These errors are specific to one operating system.

Error messages are very descriptive that in most of the cases the errors can be debugged without going through the manuals.

ORACLE's extensive self-checking helps to detect RDBMS internal errors ( when a process meets an unexpected condition). It issues a catchall error message for ORACLE program exceptions in the following format :

ORA-600 internal error code, arguments [num], [?], [?], [?], [?].

*where the message text is followed upto six arguments which indicates the origin and the attributes of the errors. The first argument is the internal error code number and the other arguments are various numbers, names and character strings. ( Empty brackets may be ignored ).*

Such bugs can be debugged by reporting as a S/W bug to CUSTOMER SUPPORT with all the six arguments.

When an error occurs while in recursive routine, instead of issuing the same error message for each recursive step, ORACLE displays what is happening by adding 10000 to the error number of the last recursive error message.

Trace files can also be used to debug errors. Each time an ORACLE instance is started or an unexpected event occurs in a user process or in a background process, a trace file is created with the file extension as TRC and the file name includes the process name, instance name and the ORACLE process number. The contents of the file may include the dumps of the system global area, process global area, supervisor stack and registers.

The location for trace files created by the ORACLE background process ( PMON, DBWR, LGWR, SMON ) and user processes ( SQL*PLUS, PRO*C ) can be known by the INIT.ORA parameters viz BACKGROUND_DUMP_DEST and USER_DUMP_DEST.

These trace files has to be formatted (using DUMPFMT utilility ) before reporting to CUSTOMER SUPPORT to help solve the problem.

# 4. LAB MANAGEMENT SYSTEM

## 4.i. INTRODUCTION

In any S/W organization, all the S/W developed has to undergo a complete cycle of the following processes:

- Laying out specifications
- Designing
- Implementation
- Self testing
- Regression testing
- Release

The testing phase is generally best carried out in an environment which is almost same to the actual real time environment in which the software has to actually run. The real time environment with the associated H/W and other accessories is generally kept in labs as it is not possible to provide each individual with an individual real time work station vis-a-vis one terminal per person.

In C-DOT, the entire S/W is developed for the electronic switching system, C-DOT DSS, and for testing models of the switch which are kept in labs. However there is forever a rush of designers wanting to use these models for their testing. So some sort of methodology has to be worked out for optimum utilization of the work stations by different designers on a time-sharing basis.

## 4.ii. OVERVIEW OF THE EXISTING LMS :

To ensure a proper utilization of work stations the designers are allotted time slots on different Work stations in 2/3 hr. shifts. The shift-incharge gets requests from different designers for a shift and according to the priorities, he makes a shift plan for all work stations. The shift plan is circulated to all the concerned engineers. This is how the present LMS is functioning in C-DOT.

## 4.iii. NEED FOR THE AUTOMATION OF THE LMS

However this approach although appears to be good, seems to have many flaws. To mention a few :

- Invariably designers find the system down during their shift time such that most of their time is spent in bringing it to a workable condition.
- Shifts may be allotted on work stations which are not according to the required configuration. So there will be wastage of time in terms of accommodating all desired resources and getting them fixed
- Non availability of information regarding the state in which work stations have been left by previous users.
- Non availability of the information regarding the patches installed or removed in a particular work station.
- No one directly held responsible for all the chaos generally encountered and hence it continues.
- Non availability status of the resources along with the work stations unknown till the time of the allotment of shifts.
- Cancellation and reallocation of the shifts can be done only by the shift incharge. It is a risksome on the part of the engineer to approach shift incharge for even a simple task like shifts cancellation and reallocation.

The only way all these discrepancies could be removed was by automating the entire system to the maximum extent possible.

## 5. AUTOMATED LAB MANAGEMENT SYSTEM

The drawbacks of the unautomated lab management system can be overcome by automating the system.

By automating the LMS, ws mean the following should b provided:

1. Users should be allowed to give inputs, lik requesting for shifts and etc, online without the interventio of the LAB MANAGER.

2. Lab manager need not bother about the scheduling o the shifts unless in some extreme conditions, like when th scheduling done by the system needs some alterations and etc.

3. User should be allowed to select the shift according to his convenience and shifts availability.

4. Users should be known of the current softwar status of the labs. For future reference we shall call ther Patch/Prom links.

5. The estimation of the utilization of the labs should be done without much difficult.

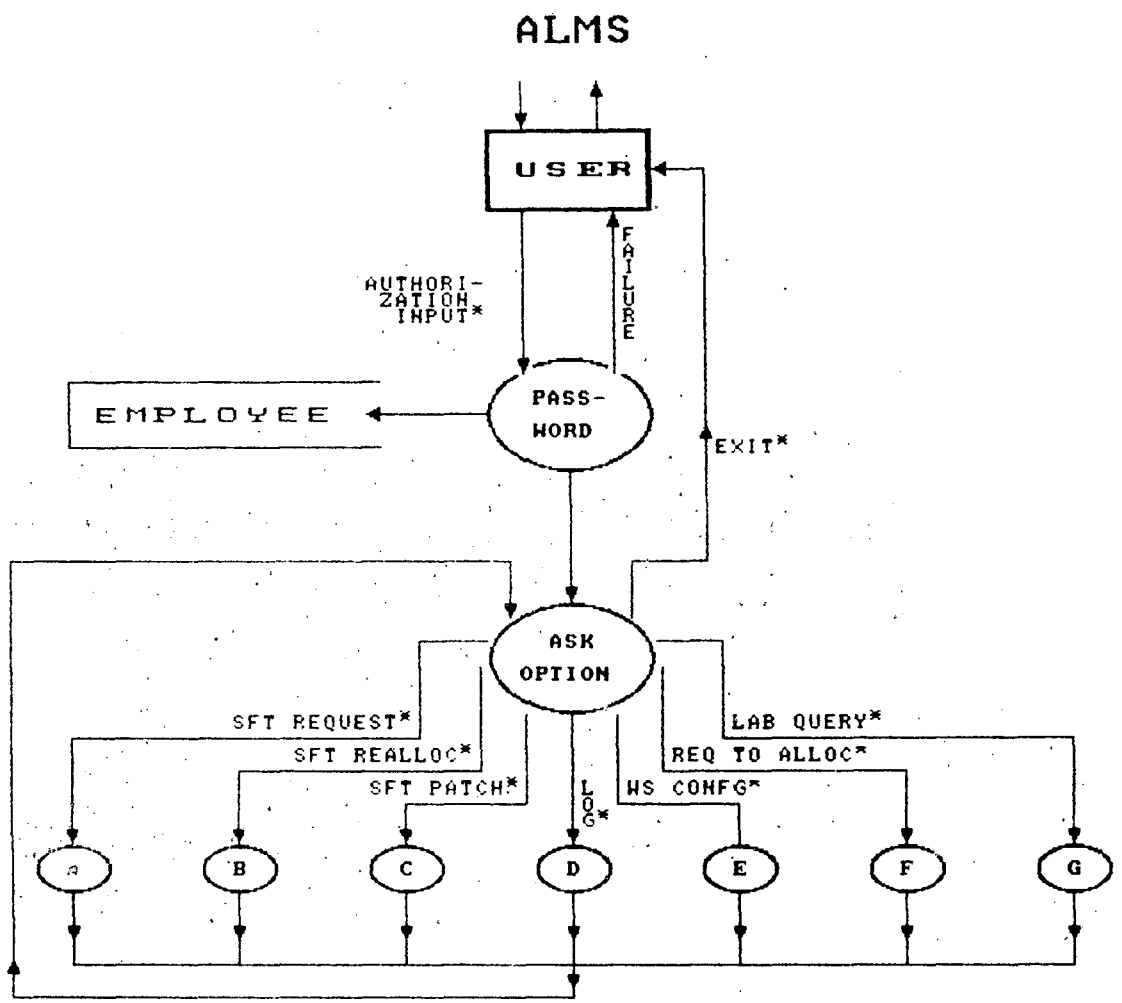
6. Data security should be provided so that any user other than lab managers, can access only his records and nc one else's.

In this section, the outline of the AUTOMATED LAB MANAGEMENT system is described.

*The approach and the notation employed for the discussion of the development of the application program, ALMS is considered here.*

*The whole ALMS process is represented by one data flow diagram (DFD). Thereafter each process and sub process of every DFD is described either algorithmically in a few steps or through a DFD again, which ever method is more convenient. For each DFD the contents of the data items that pass along the arcs of the DFDs are described. Finally the implementation of each process on ORACLE is discussed.*

*Notation used while drawing the DFDs are same as the standard notation of the database DFDs. Each bubble represents a process. Data base tables are represented by a rectangle with only three sides shown. The source and the sink are shown by a closed rectangle. The data flows between processes, sources and sinks. The data flow is shown along the arcs.*




A ⟶ SHIFT REQUEST
B ⟶ SHIFT CANC / REALLOC
C ⟶ PATCH LINKS
D ⟶ SHIFT LOG
E ⟶ WS CONFIGURATION
F ⟶ SHIFT STATUS CHANGE
G ⟶ LAB QUERY

---

* TO BE INPUTTED BY THE USER.

## i. DESCRIPTION AND IMPLEMENTATION OF ALMS

DESCRIPTION :

The DFD for the whole ALMS shown in the previous page is implemented in several forms as described below.

Each (sub)process in the ALMS DFD, more or less, is implemented as a separate form. There are around ten representative forms designed, each of which serves a distinct purpose.

In this section a brief description of how each process is transformed into a form and what purpose does each such form serves is given.

When the user logs in, he has to enter his correct user name along with his password (arc AUTHORIZATION INPUT). The process PASSWORD checks whether the user name and the password entered are valid.

If the user authorization succeeds, the PASSWORD issues an AUTHORIZATION SUCCESS status and the control goes to the next process, ASK OPTION.
However, if the user enters a wrong password , the PASSWORD gives an AUTHORIZATION FAILURE signal and the control goes back to the user enabling the user to try for a successful login this time.

This process PASSSWORD is implemented using one form. The process ASK OPTION which is run when the user makes a successful login, is implemented as a form called

ASK_OPTION.

This process reads the purpose of the user who is running the LMS and branches to the sub-process as required by the user.The process ASK OPTION may call any of the following seven sub-processes depending upon the request of the user :

A. SHIFT REQUEST :

This process is called when the user wants to make a request for the shifts. The function of this process is to read the request of the employee and to store the sufficient data regarding this request in the database table. Two forms are designed to implement this process.

B. SHIFT REALLOCATION / CANCELLATION:

This process is invoked when requests for the cancellation and reallocation task. There are broadly two distinct type of shifts that may be requested by the employees for the cancellation or reallocation.

First one is when the shifts are just requested i.e. these shifts may or may not be alloted to that employee; and secondly when the shifts are

guaranteed to be allotted. These two cases are handled adeptly by this process and ensures that a consistent data transaction takes place always.

The task of this process is done by two different forms.

C. SHIFT PATCHES:

This is another sub-process of the ALMS process and is run when ever some data has to be retrieved or inserted into the database tables regarding the PATCH/PROM links, defined in the glossary.

Only one form is built to serve the purpose of this process.

D. SHIFT LOG:

After the employee uses the shifts that are allotted to him, he has to run this process which reads the details of the shifts that have been used. Rigorous checks has to be done to ensure that the data always remains consistent. This process takes care of such checks while manipulating the data in the database.

This process is implemented using one Form.

E. WS CONFIGURATION:

This process maintains a list of WS configuration components which can be displayed for the reviewing purposes. It also keeps track of all newly

installed patches and also of all new components. It allows for the insertion and/or deletion of the required components that the employees may want to insert or delete.

This process again is carried out by one form.

F. REQUESTED TO ALLOCATED

Every once in a week, all the shifts that are requested for the next week should be given a status 'WILL BE ALLOTTED. Once the status of the shift is changed to the value mentioned above, that shift will be definitely allotted unless and otherwise it is changed by the Lab manager in some extreme conditions.

The form REQ_ALL takes care of this task.

G. LAB INCHARGE QUERY:

This process is of a great help to the lab incharge who has to arrange the WS with the components as requested by the engineer in the SHIFT REQUEST process. This process decodes the coded version of the concatenation of required components' numbers which uniquely identifies the component.

After one of the above processes finishes its task, the control goes back to user from where he can run another or the same process of his interest or make an exit.

In the subsequent pages we shall see in detail, how each process with DFD, is handled by the SQL FORMS and SQL REPORTS and hence how the automation of the Lab Management is achieved.

PROCESS NAME : PASSWORD

PURPOSE : To make an authorization and to achieve the data security

DFD : Not shown.

DESCRIPTION : This process verifies whether the user is authorized user or not.

* Read and validate user name
* Read and validate password
* Alter the password if required by the user and Validate the altered password.
* Save the value of employee's number.

IMPLEMENTATION :

This process is built as one SQL*FORM with three blocks.

* A field is created in one of the blocks of this FORM to read the user name. A post change trigger is defined on this field which ensures that the name entered is in the list of C-DOT EMPLOYEES table. If an invalid username is entered, then this trigger fails and the error message is displayed. If a valid user name has been entered, the next step is executed.
* Another field is defined in the same block as the USERNAME field is defined, to read the password of the user. Similar

checks are done as in the USER NAME field. If a right password has been entered the POST-CHANGE trigger succeeds and then the next step is continued else the error message is displayed to the user.

* In the next block, the user has to enter his option whether he wants to change his password or not. If the password alteration is not required then the next process ASK OPTION is called.

However if the password has to be altered, the user has to enter the new password, in the third block. Then he has to re-enter the new password for validation. A KEY-OTHERS trigger defined on this VALIDATE PASSWORD field checks whether the contents of the NEW PASSWORD and VALIDATE PASSWORD fields match. If it matches, then the old password is replaced by the new one in the database table and ASK OPTION is called. Else the message that the validation has failed is displayed and the old password is resumed. Then the ASK OPTION process is called.

* Before going to the ASK OPTION process, the value of the employee's number is stored in the global variable, EMP_NMBR.

PROCESS NAME : ASK OPTION

PURPOSE : To process the employees' purpose for running this package.

DFD : Not shown.

DESCRIPTION :

* Read option number
* Check for the validity for the entered option number.
* Store option number for reference in future.
* Branch to the process as requested by the employee.

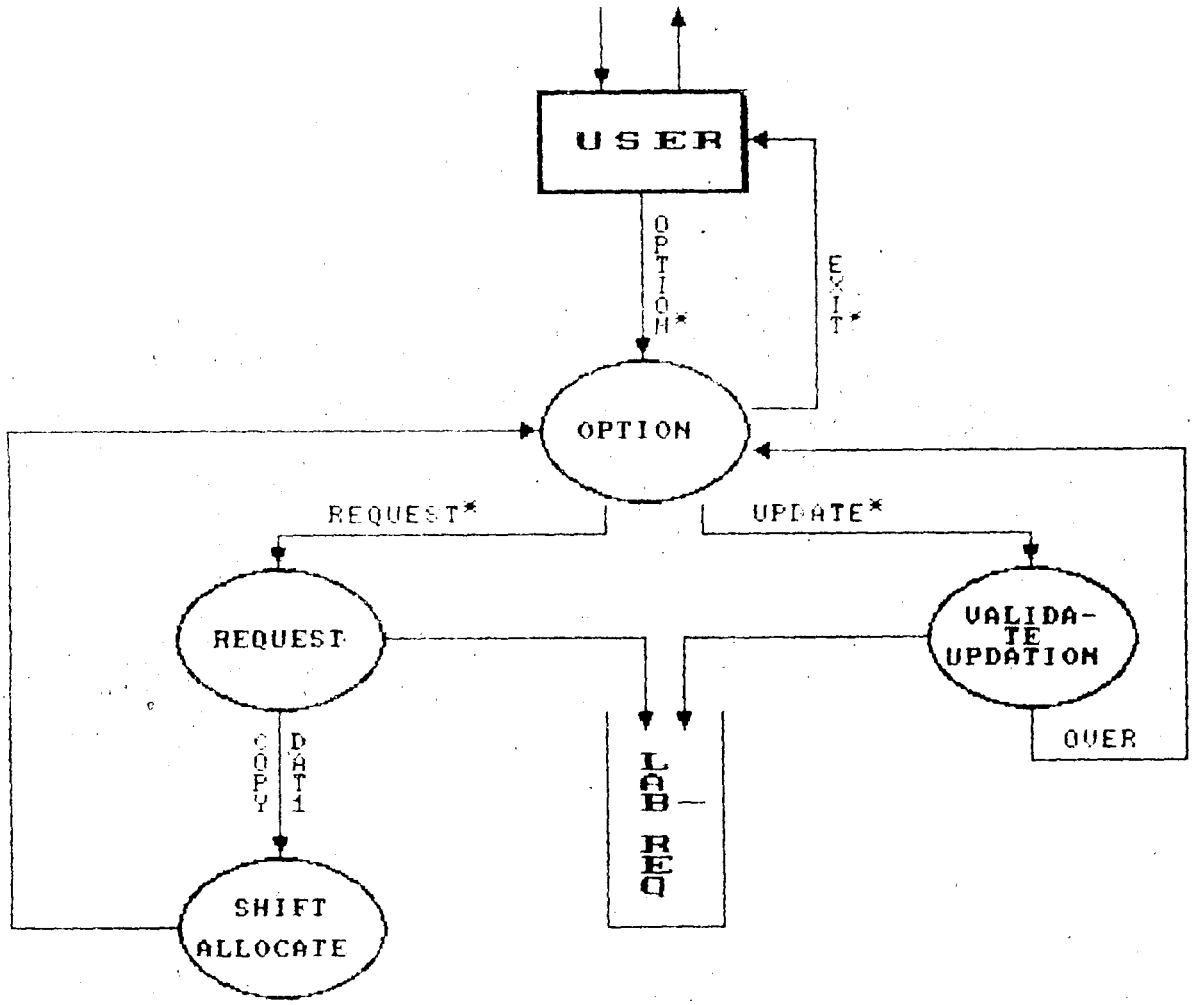
IMPLEMENTATION :

One SQL*Form with one block is created for the implementation of this process. The whole form fits in just one page.

* A field called OPTION _NO is created in this form. Employee can enter his option number in this field.
* A step in KEY_OTHERS trigger* ensures that the option number entered is between 0 and 7.
* A step in KEY-OTHERS trigger defined on this field copies option number as entered by the employee into the global variable, global.option_no

* A step in the KEY-OTHERS trigger calls the SQL*Form as requested by the employee.

# A. SHIFT REQUEST




---

* TO BE INPUTTED BY THE USER.

PROCESS NAME : SHIFT REQUEST

PURPOSE : To save and process the shifts requested by the employee.

DFD : Shown in the previous page.

IMPLEMENTATION : Two forms with 7 to 8 blocks and several triggers, validation checks etc are created to serve the purpose of this process and hence its child processes.

We shall see now the description and the implementation of each child process of the SHIFT REQUEST process.

SUBPROCESS NAME: GET REQ

PURPOSE : Gets the verified request details from the employee.

DFD : Not given.

DESCRIPTION :

* Read the PURPOSE for requesting for the shifts, TOTAL SHIFTS required and a concatenation of the work station configuration components that the employee requires.
* Check for the validity of the inputted data.
* Generate a unique REQUEST NUMBER for this

request which uniquely identifies this request.

* Send the data item, copy dat1, to the SHIFT ALLOCATE sub process, which is described later in this section.

IMPLEMENTATION : The task of this process is done by few triggers and validation checks. No separate SQL*FORM is created to implement this process.

* The parent SQL*Form, SFT_REQ has three INPUT-ALLOWED fields for the three variables, PURPOSE, TOTAL SHIFTS AND WS CONFIGURATION..
* A validation check is made for the TOTAL-SHIFTS field to ensure that the value entered will always be in the range 1 to 24, as there are 24 one hour shifts per day.

  A POST-CHANGE trigger is created in the CONFIGURATION field. this trigger makes sure that the first character of the configuration string always holds a valid WS type, i.e the first character should always be either S, if an SBM is requested or M if an MBM is requested.

* A PRE-INSERT trigger updates the maximum request number used so far stored in the database table SEQNOS by one and assigns this number as the request number to this particular request.
* A KEY-COMMIT trigger is defined in several steps such that after committing the transaction, the value of the REQUEST NUMBER and the WS CONFIGURATION string are copied into global variables for future references then the control is transferred to the SHIFT ALLOCATE sub process.

SUBPROCESS NAME: VALIDATE UPDATION

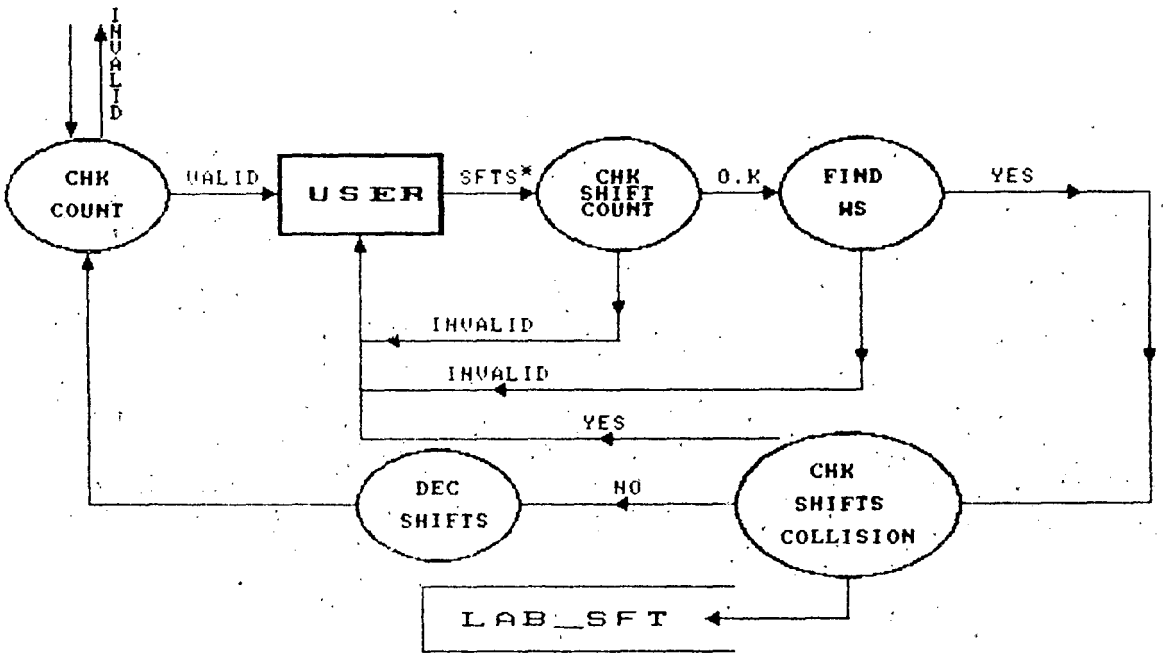
PURPOSE : Verifies that the updation done on the records is valid.

DFD : Not shown.

DESCRIPTION :

* Display only those records to which the employee is allowed to do updation process.
* Allow updation only to the PURPOSE field and the WS CONFIGURATION field. In the later case the first character of the string is not restricted from updation as it contains the type of the work station the engineer wants to use the shifts in.

# SHIFT ALLOCATE




---

* TO BE INPUTTED BY THE USER.

LEMENTATION: Even this sub-process is implemented in ORACLE through triggers, validation checks and usir[illegible].

* A WHERE clause in the WHERE / ORDER BY option window is written so as to retrieve at the time of execution of the query only those records with employee number same as that stored in the global variable.
* For total shifts field, the UPDATION ALLOWED option is deselected in the ATTRIBUTES WINDOW.
* A PRE-UPDATE trigger is defined to check that the first character of the WS CONFIGURATION field is not changed as this first character contains the type of the WS requested.

?ROCESS NAME: SHIFT ALLLOCATE

?OSE : To provide on-line selection of the shifts.

: Shown in the previous page.

:RIPTION :

* CHK COUNT

  If the shifts to be chosen is zero then this sub-sub-process issues a 'ALL SHIFTS CHOSEN' message and terminates its parent process i.e. SHIFT ALLOCATE.

However if the shifts to be chosen is greater than zero then this process issues the message 'SHIFTS ARE TO BE CHOSEN' and the control goes to the user to read the input.

* CHK SHIFT COUNT

Read shift details viz on which date what shifts are required.

If the SHIFTS TO BE CHOSEN is greater than or equal to shifts that are requested at present, then the process invokes the next process else the control goes back to the user with the message ' TRIED TO CHOOSE MORE SHIFTS'.

* FIND WS

If any of the Work Stations (WS) of the type as chosen in the SHIFT REQUEST process, has free shifts on the date and during the shifts as chosen by the user in the above process, and can accommodate all the configuration components as requested by the user for these shifts then a status signal 'VALID' is issued and the next process,SHIFTS COLLISION starts its execution.

* CHK SHIFT COLLISION

  If any interactive shifts* are chosen by the user and if few or all of these interactive shifts has already been chosen by the same user on the same date the this process issues a 'SHIFTS COLLISION' signal and the control goes to the user else the control goes to the DEC SFTS process.

* DEC SHIFTS

  This process decrements the value stored in the SHIFTS TO BE CHOSEN variable by the number of shifts that are chosen.

IMPLEMENTATION :

The subprocess SHIFT ALLOCATE is implemented in ORACLE using one form with seven blocks and several triggers. The implementation of each subprocess whose description is given above is described.

* CHK COUNT

  In the KEY-COMMIT trigger before COMMIT macro, a check is made to ensure that the contents of the field 'SHIFTS TO BE CHOSEN' is greater than or equal to zero.

* CHK SHIFT COUNT

  Three INPUT ALLOWED fields are defined in the SQL*FORM to read the values of the

shift date, start and the last shift numbers from the user.

Few steps in the PRE-INSERT trigger are written to check that the SHIFTS TO BE CHOSEN field is not less than the difference of the last and the start shifts as requested by the user.

* FIND WS

Few steps in the PRE-INSERT trigger are written to check the following:

(The trigger aborts if any of the following two steps fails and the control goes back to the approriate message else the control goes to the next process.)

Whether there are any WSs, of the type as chosen by the user in the SHIFT REQUEST process, free on the date and during the shifts as required by the user.

Whether all configuration components as required by the user are available in any of the WSs chosen in the above step.

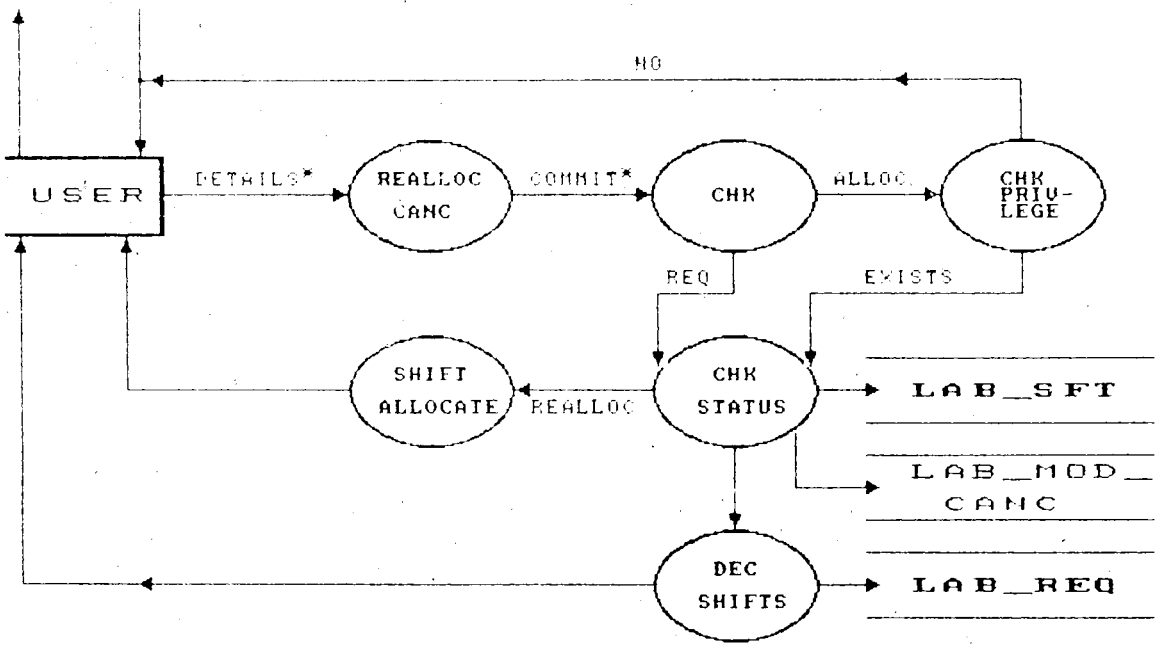
* CHK SHIFT COLLISION

A step in the PRE - INSERT trigger ensures that the same user will not be given the same interactive shifts on any date.

)EC SHIFTS

\ POST-INSERT trigger is defined so that
:ach time a record of shift details is
inserted into the table, it decrements
3HIFTS TO BE CHOSEN value by one.

# B. SHIFT CANC / REALLOC




* TO BE INPUTTED BY THE USER.

PROCESS NAME : SHIFT CANCELLATION AND REALLOCATION

PURPOSE : Provide on-line cancellation and reallocation of the requested shifts.

DFD : Shown in the previous page

DESCRIPTION : This process has six child processes.

* REALLOC CANC

  Read user's name, request number, work station number, shift date, and start and the last shift numbers to which the cancellation or reallocation is requested along with the status value for running this process viz REALLOC for reallocation or CANC for cancellation.

* CHK

  If the status of the above shifts as entered by the user is 'ONLY REQUESTED' and not 'WILL BE ALLOTTED' then this process passes the control to the CHECK STATUS process else the control goes to the CHK PRIVILEGE process.

* CHK PRIVILEGE

  Read password.

  If a right password has been entered then the process is terminated and the control goes to the next process viz, CHK STATUS

Else the process aborts and the control goes back to the user.

* CHK STATUS

If the shifts are to be canceled then the process DEC SHIFTS is invoked else the process SHIFT ALLOCATE is invoked. This process also removes all shifts that are given, either for cancellation or reallocation, from the LAB_SFT table after inserting the same into the SHIFT_MOD_CANC table which has the details of those shifts that are canceled or reallocated.

* DEC SHIFTS

This process decrements the value stored in the TOTAL SHIFTS column of the corresponding record in LAB_REQ table by the number of shift that are cancelled.

* SHIFT ALLOCATE

Same as that in the SHIFT REQUEST process.

IMPLEMENTATION :

The process, SHIFT REALLOCATION /CANCELLATION is implemented through two SQL*FORMS with around eight blocks and several triggers. Implementation of each sub-process is given below which constitutes to the implementation of the whole process.

* REALLOC CANC

  This process is implemented through an SQL*Form with fields for entering the values of the variables EMP-NMBR, REQUEST NUMBER, SHIFT DATE, WS NUMBER, START AND LAST SHIFT NUMBER and the purpose of running this form.

  This form also has a field to display the total number of shifts that are cancelled or reallocated.

* CHK

  The task of this sub-process is done by a part of the PRE-INSERT trigger defined in few steps. Few steps in the PRE-INSERT trigger is defined to check that all the shifts as inputted by the user has the status as 'REQ' and not 'ALLOC'. If this part of the trigger succeeds then the process CHK STATUS is invoked. Else the process CHK PRVG is invoked.

* CHK PRIVILEGE

  This process is implemented as a BLOCK of the SQL*FORM. It has the field called PASSWORD with ECHO option deselected. If a wrong password has been entered the POST

CHANGE trigger defined on this field fails and control goes to the user with the failure message. Else the control goes to that part of the PRE-INSERT trigger by means of which the process CHK-STATUS is implemented.

* CHK STATUS

  This process is a part of the KEY-COMMIT trigger. This process is defined after the COMMIT macro in the trigger. It ensures that if cancellation was required then the process DEC SHIFTS is called else the process SHIFT ALLOCATE is called.

* DEC SHIFTS

  This process is defined as a user defined trigger, DEC_SFTS. It decrements the 'TOTAL_SHIFTS' column of the corresponding record stored in the LAB_REQ table (this record is identified by the request number as entered by the user) by the total number of shifts that are cancelled in order to achieve the data consistency.

* SHIFT ALLOCATE

  Same as that in the SHIFT REQUEST process.

PROCESS NAME : SHIFT PATCHES

PURPOSE : Keeps track of the information regarding the Patch/Prom links on the BMs.

DFD : Not given.

DESCRIPTION :

* This part of the process reads the purpose why the user is calling this process. It then processes the inputted value and calls the appropriate sub processes. It also validates the entered option.

  The valid options are the following :

  Report successful installation of the patch link.

  Report the successful removal of the patch link.

  Report the failure to install the patch link.

  Report the failure to remove the patch link.

  Just to know the details of the patch links.

  Update the status of the patches.

* Validation of the the inputted or updated records.

Users can update only those records with patch link status as 'FAILED TO INSTL' and 'FAILED TO REMOVE' to 'SUCCESSFULLY INSTLD' and 'SUCCESSFUL REMOVED' respectively.

Users should be allowed to perform nothing more than what has been chosen in the last sub process.

IPLEMENTATION:

The PATCH_LINK process is implemented through a single FORM having two blocks.

* In the first block the user has to enter the option number matching the action that he wants to perform..

  A KEY-OTHERS trigger is defined to check that the entered option number is valid. If the option number entered is that of EXIT then the process terminates else the next block is called.

* A post field trigger is defined in the STATUS field of the second block. This trigger checks in non update mode, before leaving that field whether the status field has the value as opted in the first block. If not, the cursor remains in the field else the next command is executed.

A POST CHANGE trigger is defined on the STATUS field again, to make sure that the status which was 'FI' can be updated to 'SI' and etc.

In the COMMIT trigger that is defined on this block, checks whether the necessary data has been inputted and deletes any superfluous data entered.

For example when the STATUS is FI (for Failed to Instal) the REASON is necessary to mention why the failure has occurred, at the same time SINCE WHEN is not necessary as the patch is neither installed nor removed.

PROCESS NAME : SHIFT LOG

PURPOSE : To keep track of the information regarding the shifts that have been used.

DFD : Shown in the previous page.

DESCRIPTION :

Four small subprocesses constitutes the SHIFT LOG process. Here the description of these three sub-processes is given which hence describes the task of the SHIFT LOG process.

* CHK STATUS

  This process processes the data entered by the user. It checks whether the shifts mentioned are really allotted to the user or not. If the shifts are allotted then the process terminates successfully and the next process, CANCEL begins its execution. However, if the user inputs the details of those shifts that are not yet allotted to him, then this process aborts transferring the control to the user along with the error message.

  CANCEL

  This process will be made active when the CHK STATUS process finishes its task successfully.

This process deletes the shifts details for which the log is being made from the LAB_SFT table.

It also stores enough information of the shifts that are being logged. It is through this information that one can estimate the utilization of the Work stations.

* DEC SHIFTS

The task of this process is mainly to keep the data consistently. This process keeps track of the shifts that are yet to be used by the employee.

IMPLEMENTATION :

The SHIFT LOG process is implemented by means of an SQL*Form having three blocks and several triggers.

Here we shall consider the implementation of each sub-process of the SHIFT LOG process.

* CHK STATUS

This task is performed by a FORM and a PRE-INSERT trigger.

The form has fields for the values of the variables to be inputted by the user.

After the user enters the shifts date,

work station number, start and last shifts the PRE-INSERT trigger checks whether the shifts as per the details given by the user has the status 'ALLOTTED' instead of 'REQ'.

If these shifts has the status 'ALLOTTED' then the next process is invoked else this process makes an abnormal termination back to the user with the error message.

* CANCEL

This process calculates how many shifts have been used. And it deletes the corresponding entry of each used shift from the LAB_SFT table.

A part of the PRE-INSERT trigger is defined as a loop as follows:

1. ss = start shift number and count = 0
2. If cur_sft > LAST_SHIFT_NO (given by the user) then go to label 4.
3. If the record in the table LAB_SFT with the values SHIFT_DATE, WS_NO, REQUEST_NMBR, EMP_NMBR as entered by the user and the SHIFT_NO is equal to cur_sft ,exists then increment the variable 'count' by one and delete the same record from the table.
4. Go to label 2.
5. END.

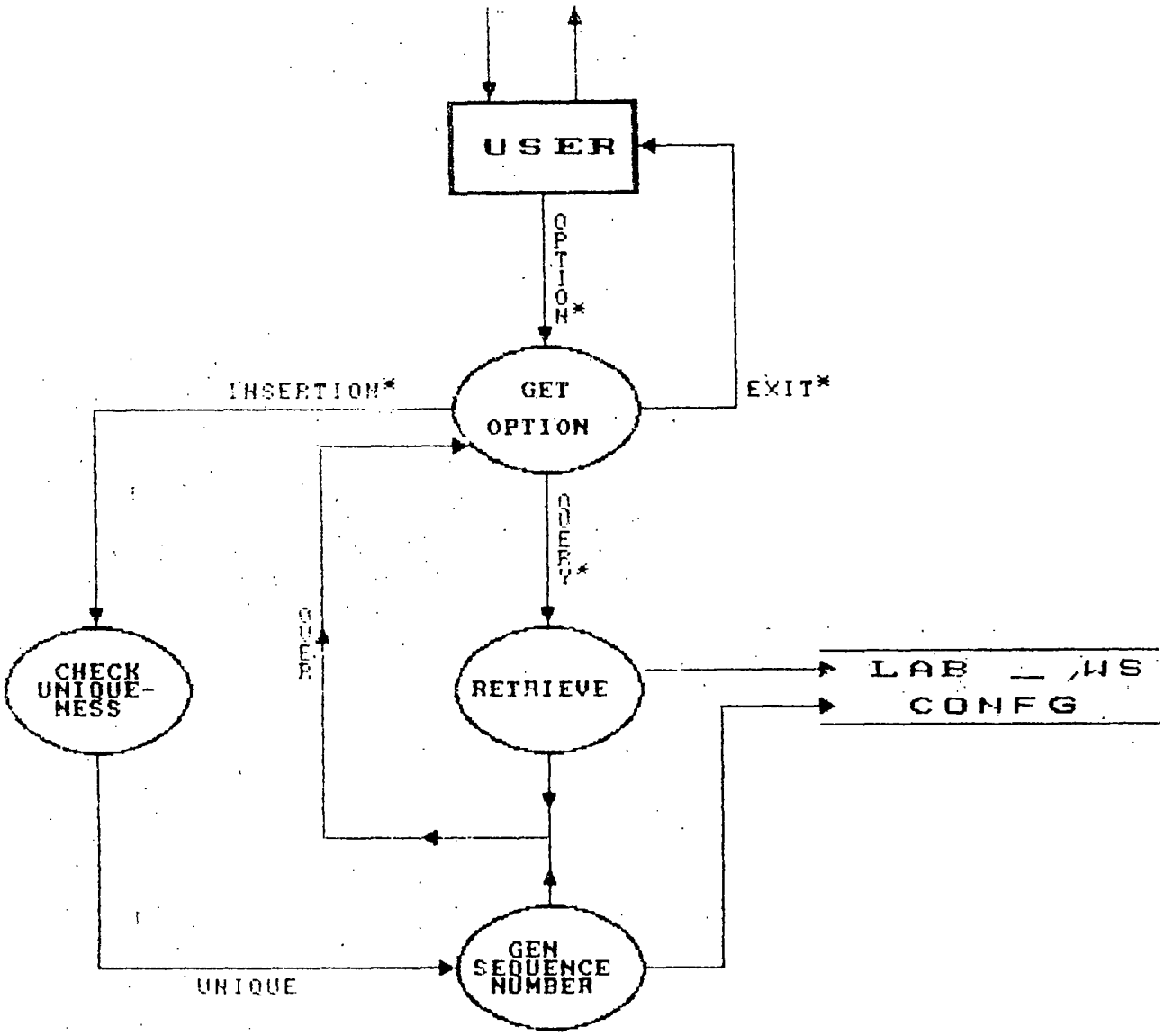
* DEC SHIFTS

This process updates the value of the TOTAL SHIFTS field of the corresponding LAB_REQ record by the number of shifts that have been used (the variable 'count' has this value.

If the TOTAL SHIFTS value is reduced to zero, then the corresponding record is deleted from the LAB_REQ table as all the shifts of that request are used.

This process is again a part of the PRE-INSERT trigger.

# E. WS CONFIGURATION




* TO BE INPUTTED BY THE USER.

PROCESS NAME : WS CONFIGURATION

PURPOSE : Keeps track of the work station configuration components.

DFD : Shown in the previous page.

DESCRIPTION :

This process has four child process, GET OPTION, RETRIEVE, CHK UNIQUENESS, GENERATE and SEQ NUMBER.

* GET OPTION

This process just checks the option selected by the user and branches to the appropriate process accordingly. If the option chosen by the user is ;QUERY' then the process 'RETRIEVE' is invoked to run the query. If the option is 'INSERT' then the process calls CHK UNIQUENESS process to ensure a valid data has been entered.

* RETRIEVE

This process is very simple. It just retrieves all the configuration components of the WS that are stored in the table.

* CHK UNIQUENESS

This process reads the component description that the user wants to store in the table.

It ensures before inserting the new component description, that the component desc as entered by the user does not exist in the table

* GENERATE SEQ NUMBER

Before the new component description is stored in the database table, a unique number has to be assigned as referring a component description is easier through a smaller parameter, like a 3 digit number, rather than a lengthy textual expression. This process generates a new sequence number which is the maximum seq number that has been used for this purpose plus one and assigns this sequence number as the component number of the new component. Then it inserts the two values, description as well as its number, into the table.

IMPLEMENTATION :

* GET OPTION

This process is implemented by using one block of the SQL*FORM. Field is created to read the user's option. A POST-CHANGE

trigger is used to process the option number as entered by the user. This trigger firstly checks whether the option number is valid or not. If the option number entered is valid and it is not QUIT then this trigger transfers the control to the next block, WS_CONFG. If the option entered is QUIT then the trigger transfers the control to the user.

* RETRIEVE

This process is a part of the KEY-STARTUP trigger in the block WS_CONFG. A CASE statement is written to process the option number as entered by the user in the previous process. If the option chosen in the previous block is 'RETRIEVE' then the EXECUTE QUERY macro is performed. As both the fields in this block are defined to be non updatable, the inconsistency is eliminated.

* CHK UNIQUENESS

This is implemented as a PRE-INSERT trigger and a part of the KEY-STARTUP trigger. If the option entered is 'INSERT' the CASE statement written in the KEY-

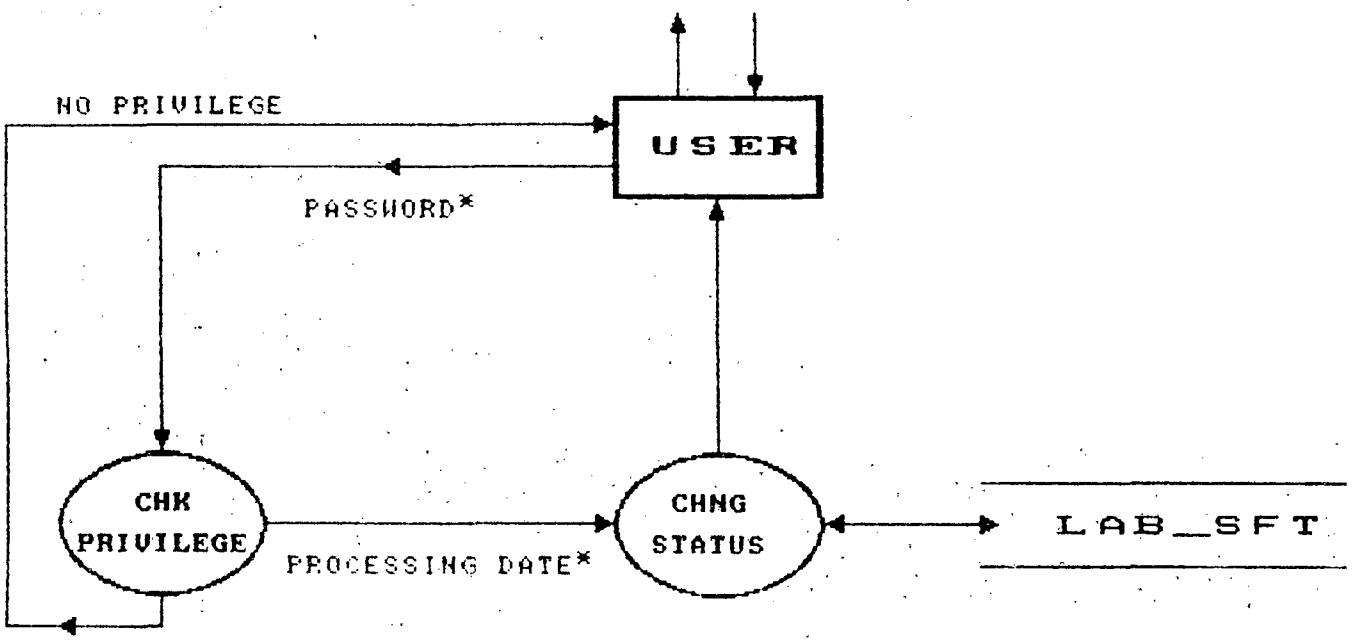
STARTUP trigger detects and the macro CREATE RECORD is performed in which the user can enter the record he wishes to commit.

After reading a new component description from the user, and after the user presses 'COMMIT' key the PRE-INSERT trigger defined in this block ensures that the new component description does not already exist in the table. If this part of the trigger succeeds then the next part of the trigger is executed (next process) else the trigger fails and the insertion of this duplicate record is not done.

* GENERATE SEQ NUMBERS

A table SEQNOS stores in a record, the maximum number that has been used so far as the component number of this table. A step in pre insert trigger retrieves this number , increments it by one and assigns this new maximum number to the new component before putting back the new maximum number into the SEQNOS table.

# F. CHANGE SHIFT STATUS




* TO BE INPUTTED BY THE USER.

PROCESS NAME : REQUESTED TO ALLOCATED

PURPOSE : To process the requests of all the employees who had asked for the shifts for next week.

DFD : Shown in the previous page.

DESCRIPTION :

This process is run once in a week to change the status of those shifts that are to be allotted in the next week from 'REQUESTED' to 'WILL BE ALLOTTED'.

This process has two child processes whose description and implementation is discussed below.

* CHK PRIVILEGE

  This process checks whether the person who is running this task is allowed to do so. If privilege exists to that person then the next process is called else the control goes to the user.

* CHNG STATUS

  This is the key task of the process, REQ TO ALLOC. The CHNG STATUS first reads the starting date of the week the employee wants to change the status. It makes a validation check to see that the date entered is not less than the current date.

Then the process updates the status of all the shifts details for which the SHIFT DATE falls in the week in question.
After completion, as other processes , it returns to the user.

IMPLEMENTATION :

This process is implemented using one SQL*FORM having four blocks.

* CHK PRIVILEGE

  A PRE-FORM trigger is defined to check that the employee's number (stored in the global variable at the time of running PASSWORD process) is in the database table LAB-MNGRS which contains a list of employee's numbers who can run this FORM. If the employee's number exists in the table, then the next process is called else the trigger fails and the control is returned to the user.

* CHNG STATUS

  A block PROC_DATE with one field, PROC_DATE is created. The user has to enter the starting date of the week for which the shift processing is required.

A post change trigger is defined in this field to ensure that the user enters the date that comes after the current date.

A PRE-UPDATE trigger is defined on the field, PROC-DATE. This trigger retrieves all the shifts details which has the SHIFT_DATE value between the date as entered in the PROC_DATE field and PROC_DATE plus 6. It then changes the status of each retrieved record from 'REQUESTED' to 'WILL BE ALLOTTED'.

PROCESS NAME : LAB QUERY

PURPOSE : To aid the lab-in-charge in setting up the WS configuration components as requested by each employee.

DFD :

DESCRIPTION :

* Ensure that the employee who is running this form has privilege to do so.
* Read the date and the work station number for which the user wants to see further details.
* Retrieve all the shifts details that are allotted on the date and in the work station as entered by the user.
* Retrieve the request details of that shift as chosen by the user.
* Decode the encoded version of the configuration string.

?LEMENTATION :

* A PRE-FORM trigger is defined to check that the employee's number (stored in the global variable at the time of running PASSWORD process) is in the database table LAB-MNGRS which contains a list of employee's numbers who can run this FORM.

If the employee's number exists in the table, then the process execution is continued else the process is aborted and the control is transferred to the user.

* A block, DATE_WS,with two fields, one for DATE and another for WORK STATION number is created. The user has to enter the date and work station of his interest.
* A KEY-NXTFLD trigger is defined on the last field of the DATE_WS block to perform the following:

  Go to the block, SHIFT DETAILS, where the retrieved records of the shift details can be displayed.

  Perform the execution of the query with the WHERE clause as 'WHERE SHIFT_DATE = :DATE_WS.SHIFT_DATE AND WS_NO = :DATE_WS.WS_NO' and order clause as 'ORDER BY SHIFT_NO ASC'.
* A KEY-ENTQRY trigger is defined to go to the block 'REQUEST_DETAILS' and to execute a query there with REQUEST NUMBER= :SHIFT_DETAILS.REQUEST_NUMBER. So whenever the user presses SELECT key with the cursor on the shift number of his

interest, this trigger retrieves the details like ,to whom this shift has been allotted, what configuration did he request and etc.

* A KEY-HELP trigger is defined to decode the concatenated string of component numbers. This trigger saves the string till it finds the separator (I have used a + sign as a separator between two components numbers). It then starts from this separator and again saves the string till it finds another separator. This process is continued till the end of the string is reached.

Then, this trigger gets the description of each component number saved, from the database table WS_CONFG which stores the workstation component along with its identity number.

CONCLUSION:

BROAD WORKING MECHANISM OF THE ALMS PACKAGE:

Each day is divided into 24 shifts of one hour duration each.

User gives his request online for the allotment of shifts.

Shift are chosen at the time of giving request.

Shifts will be allotted in only those work stations where the components required for the engineer who wants to work on those shifts are available.

The status of each shift is 'REQ' at the time of requesting for the shift.

The status of the shifts will be changed on every FRIDAY for those shifts which fall in next week.

User can cancel or request for reallocation only those shifts which have the status 'REQ'.

Lab manager has privilege to cancel or reallocate any shifts.

User inputs after using the shifts allotted to him. These inputs aid to estimate the lab utilization.

Lab incharge will be displayed of the components that any particular engineer has requested.

In the next section, designing of the database tables and its normalization issues are considered.

ii. TABLES DESIGNED AND NORMALIZATION DONE

TABLES DESIGNED

Around ten relations are designed for various purpose. In this section, each table is mentioned briefly. The details of each table is given in appendix A.

1. EMPLOYEE

This table has the information about each employee in the organization. The details like the employee's name, his number, password and etc are stored in this table.

2. LAB_REQ

This table stores the information about each pending request. Once the request is fulfilled, the corresponding details about that request is deleted.

3. LAB_SFT

It stores the details about each shift requested including the data, ws number, shift number. Records will be deleted if the corresponding shifts are used.

4. LAB_SFT_CANCEL

It stores the complete history of the canceled shifts and those shifts which are reallocated.

5. LAB_PATCH_LINK

Has the information of the private patches. Whether these patches are installed, removed or to be installed and etc can be known through this table.

6. LAB_SFT_LOG

It keeps a history of all those shifts that have been allotted. This relation is very useful in estimating the utilization of the work stations.

7. LAB_WS_DEF

This relation has the definition details of each work station, like the type of the work station, its location, and etc.

8. SEQNOS

This table saves the maximum sequence numbers that has been used for a particular purpose like generation of REQUEST NUMBER.

9. LAB_CONFG

It has the list of all the components that are available in the work stations. It even maintains the information about the private patches.

NORMALIZATION ISSUES:

Database tables should not involve redundancy Normalization theory is applied for the design of database tables. All the tables designed are atleast in BOYCE / CODD normal form (BCNF). Here only few tables are considered and are proved that they are in Boyce Codd Normal Form. For the rest of the tables, it can be proved on similar lines.

A relation is said to be in BCNF if and only if every determinant is a candidate key.

Consider the LAB_REQ relation. Here there is only on candidate key, REQ_NUMBER which uniquely identifies the tuples. Clearly this relation is in BCNF because the primary key is the only determinant in it.

Considering LAB_CONFG, it is in BCNF because each of the two attributes that the relation has, is a candidate key and hence on each candidate key the other candidate key is fully functionally dependent which satisfy the necessary and sufficient condition for a relation to be in BCNF.

LAB_SFT is also in BCNF. There is only one candidate key (with four attributes viz REQ_NMBR, EMP_NMBR, SHIFT_DATE, WS_NO).Each of the other attributes are fully dependent on the candidate key and not on any subset of the candidate key.

Similarly the rest of the relations can also be proved to show that they are in BCNF.

iii.HOW USER FRIENDLY THE PACKAGE IS ?

The user who is using the package need not learn its internal structure, like how it is built, how the system processes his inputs and the like. A package is said to achieve the nature of an UFI (User Friendly Interface) even if the novice user could run the package without facing any problems.

In this section we shall discuss how user friendly the package is. The following are provided to achieve the friendly interface between the user and the system:

o Help is provided to all the SQL*Forms developed, through which the user can find out the information about the FORM that he is running. This information includes the function of the FORM, the purpose of each field created in the FORM, and in which fields the values are to be inputted.

o A HELP TEXT is provided at the end of each FORM . This text contains the information about primary keys that are used in that FORM. This help can be used by the user for a quick review of the functioning of the keys.

o Same convention is employed in all the FORMS while creating. So, a user will not face any confusion while running different FORMS as the keys that serve a particular purpose in one FORM serves the same purpose in all the other FORMS.

- Automatic help is provided for all the fields in the FORM. When the user moves the cursor into any field, a help of what is expected to be inputted into the field is displayed.

- Rigorous checkings are made in order to achieve a good consistency of the data, before transactions are committed into the database table .

- Users can select the required shifts on-line. He can as well know on which date what shifts are not available which would help him select the available shifts.

## iV. FUTURE ENHANCEMENTS

The major improvements that can be considered in the second phase of the ALMS development are the following:

- o Reports can be generated for all relations.
- o Change over shift should be provided during which lab incharge, who sets up the configuration in the work station, can modify the existing configuration to match the needs of the next engineer.

## 6. SHORT COMINGS OF ORACLE

While working on ORACLE RDBMS, few drawbacks of the system are noticed. To mention a few :

Though indexing minimizes the retrieval time of the results of a query, the more the number of indexes the more diffucult to maintain the table by the system.

While using SQL*Forms tool, it does not allow the transfer of triggers from one level to another.

Global variables cannot be reffered directly by SQL statements.

In many cases function keys of the SQL*Form donot match with that of the SQL*Report for the same function.

## BIBLIOGRAPHY


1. *An Introduction To Database Systems*

   ***C.J.DATE***

2. *Database Processing: Fundamentals Design, Implementation*

   ***DAVID KROENKE***

3. *A Guide To INGRES*

   ***C.J.DATE***

4. ***ORACLE Manuals***

# APPENDIX A

## TABLES DESIGNED

[Underlined attributes are primary keys.]

LAB_EMP:

| Attributes | Null? | Type |
|---|---|---|
| EMP_NMBR | NOT NULL | NUMBER |
| ENAME | NOT NULL | CHAR(20) |
| PWD | NOT NULL | CHAR(20) |

PASSWORD:

| Attributes | Null? | Type |
|---|---|---|
| PASSWORD | NOT NULL | CHAR(20) |
| DESIGNATION | NOT NULL | CHAR(20) |

LAB_REQ:

| Attributes | Null? | Type |
|---|---|---|
| REQ_NMBR | NOT NULL | NUMBER(4) |
| DATE_OF_SUBMISSION | NOT NULL | DATE |
| TOTAL_SHIFTS | | NUMBER(2) |
| PURPOSE | | CHAR(100) |
| CONFIGURATION | | CHAR(200) |
| EMP_NMBR | NOT NULL | NUMBER(4) |

LAB_SFT:

| Attributes | Null? | Type |
|---|---|---|
| REQ_NMBR | NOT NULL | NUMBER(4) |
| SHIFT_DATE | NOT NULL | DATE |
| SHIFT_NO | | NUMBER(2) |
| WS_NO | NOT NULL | NUMBER(2) |
| EMP_NMBR | | NUMBER(4) |
| STATUS | | CHAR(1) |

LAB_SFT_CANCEL:

| Attributes | Null? | Type |
|---|---|---|
| REQ_NMBR | NOT NULL | NUMBER(4) |
| SHIFT_DATE | | DATE |
| SHIFT_NO | | NUMBER(2) |
| WS_NO | | NUMBER(2) |

| | | |
|---|---|---|
| EMP_NMBR | | NUMBER(4) |
| CANCELLED_OR_REALLOCATED | NOT NULL | CHAR(1) |
| DATE_OF_CANCELLATION | | DATE |
| REASON | | CHAR(20) |

LAB_WS:

| Attributes | Null? | Type |
|---|---|---|
| WS_NO | NOT NULL | NUMBER(2) |
| SHIFT_DATE | | DATE |
| START_SHIFT_NO | | NUMBER(2) |
| LAST_SHIFT_NO | | NUMBER(2) |
| DOWN_TIME | | CHAR(5) |
| PROBLEMS_FACED | | CHAR(100) |
| EMP_NMBR | NOT NULL | NUMBER(4) |
| TOTAL_SHIFTS | | NUMBER(2) |
| SYS_CHANGES | | CHAR(150) |

LAB_WS_DEF:

| Attributes | Null? | Type |
|---|---|---|
| WS_NO | NOT NULL | NUMBER(2) |
| WS_TYPE | | CHAR(3) |
| ROOM_NO | | NUMBER(4) |
| RELEASE | | CHAR(15) |
| SINCE_WHEN | | DATE |

LAB_PATCH_LINK:

| Attributes | Null? | Type |
|---|---|---|
| WS_NO | NOT NULL | NUMBER(1) |
| PATCH_OR_PROM | NOT NULL | CHAR(20) |
| STATUS | | CHAR(2) |
| SINCE_WHEN | | DATE |
| REASON | | CHAR(50) |
| EMP_NMBR | | NUMBER(4) |

LAB_CONFG:

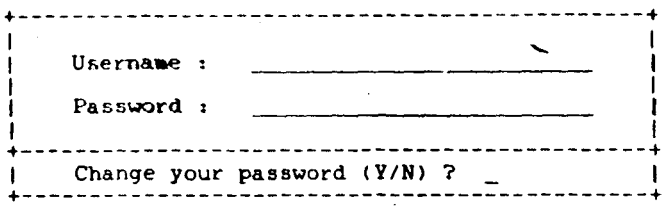
| Attributes | Null? | Type |
|---|---|---|
| COMP_NO | NOT NULL | CHAR(4) |
| DESCRIPTION | NOT NULL | CHAR(100) |

SEQNOS:

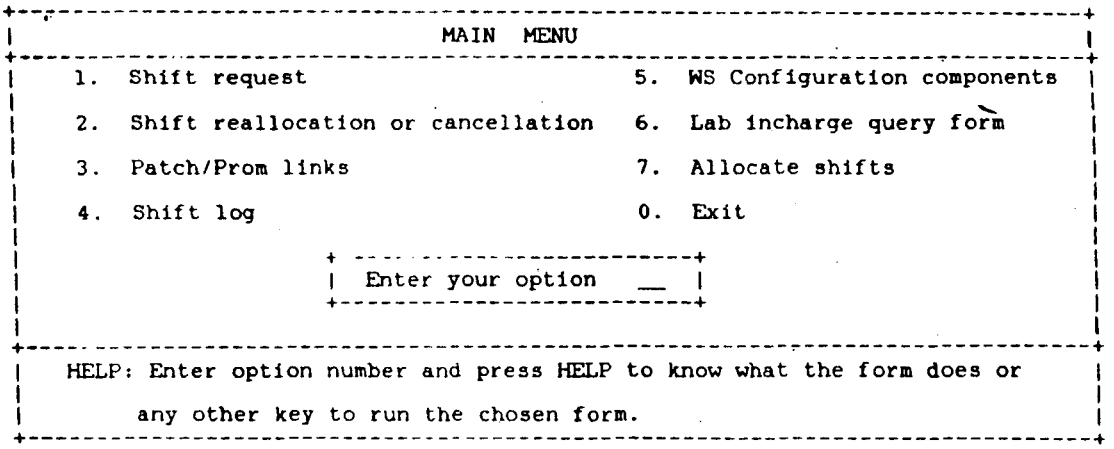
| Attributes | Null? | Type |
|---|---|---|
| BLOCKNAME | NOT NULL | CHAR(30) |
| MAXSEQNO | | NUMBER(3) |

# APPENDIX B

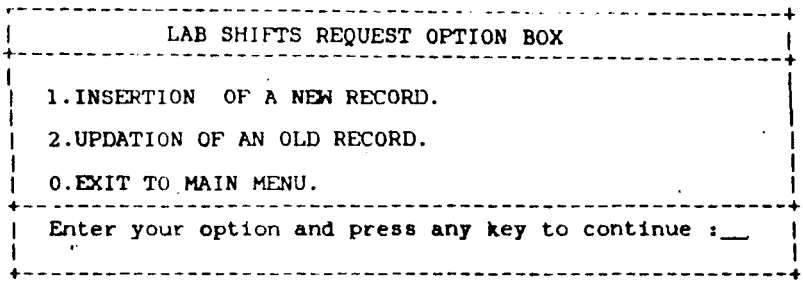
LAB MANAGEMENT



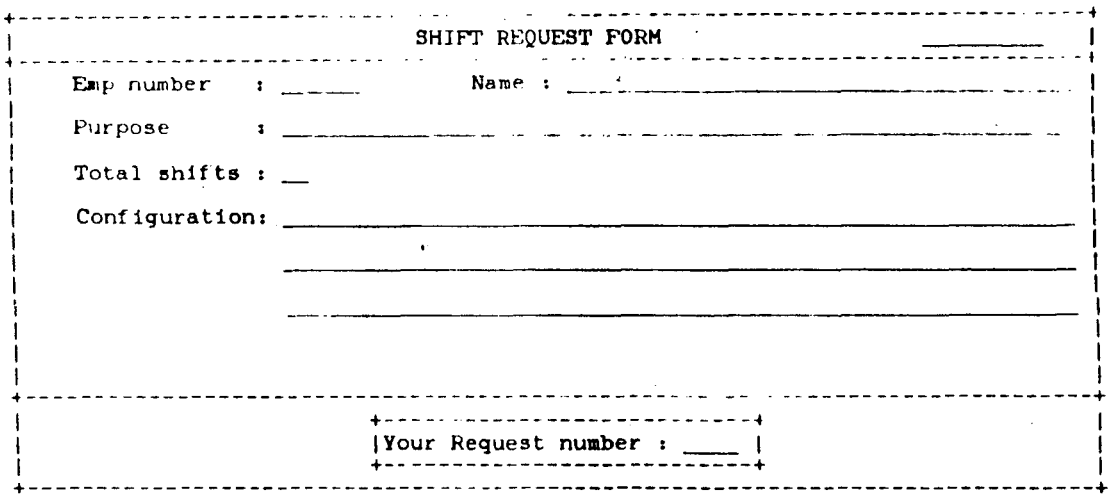

Form: lms Block: PW Page: 1 SELECT: Char Mode: Replace
MVAXA>



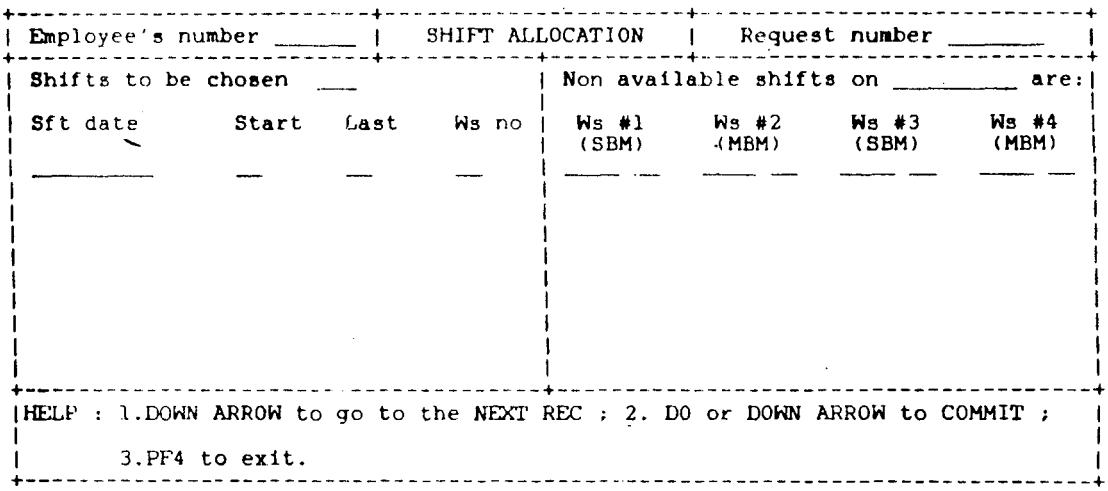

Form: OPTION Block: option Page: 1 SELECT: Char Mode: Replace
MVAXA)



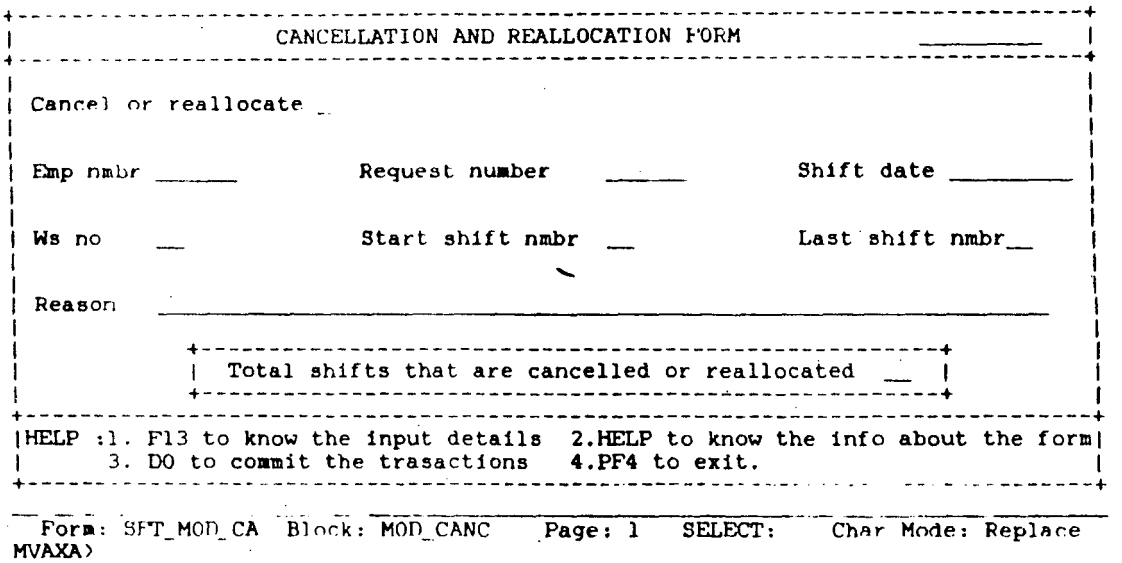

Form: SFT_REQ Block: ask_option Page: 1 SELECT: Char Mode: Replace
MVAXA)



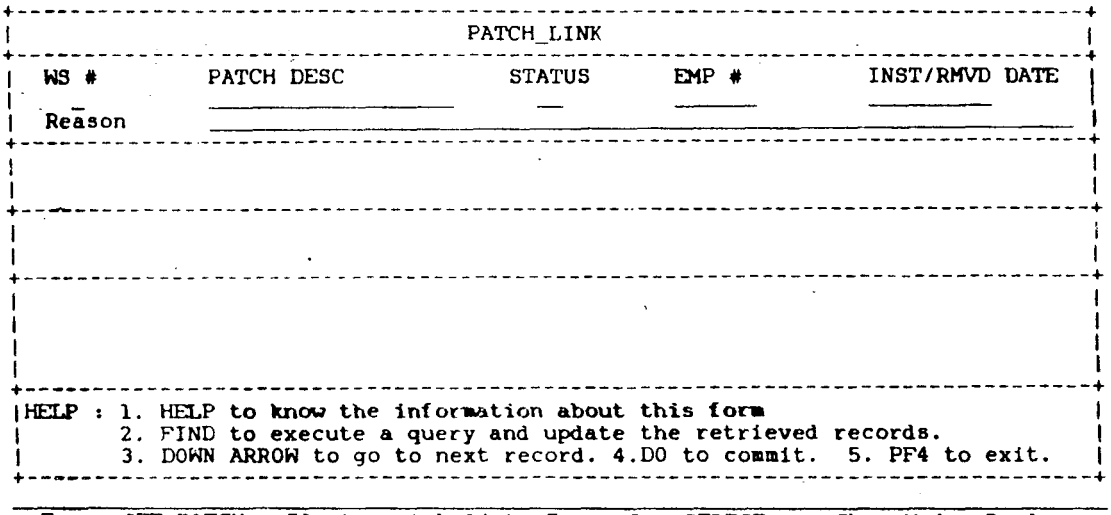

Form: SFT_REQ Block: request Page: 2 SELECT: Char Mode: Replace
MVAXA>

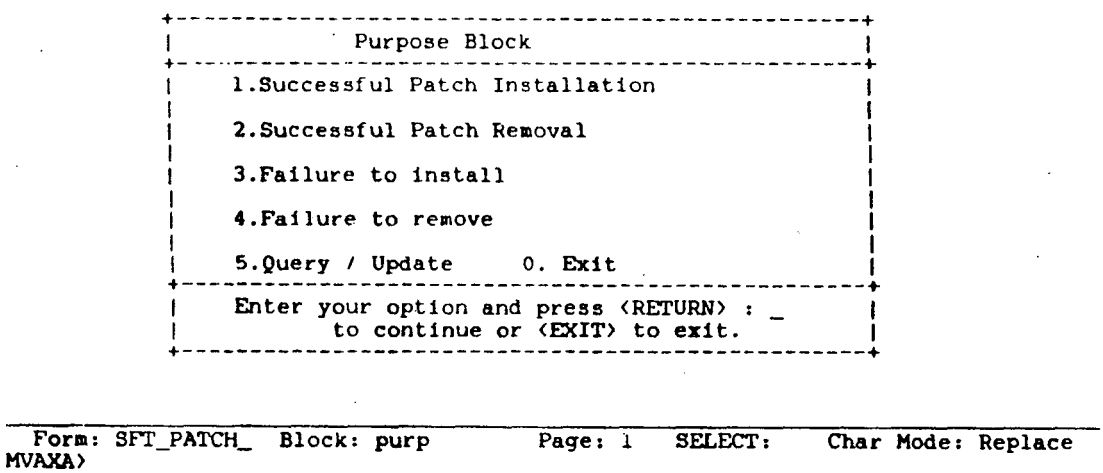


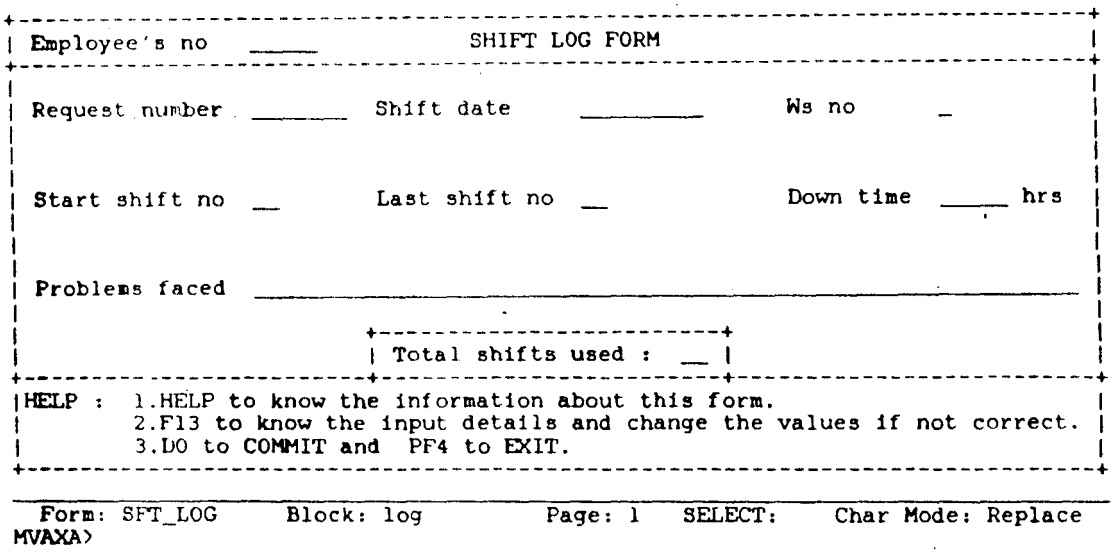

Form: SFT_ALLOCA Block: allocate Page: 1 SELECT: Char Mode: Replace
MVAXA>



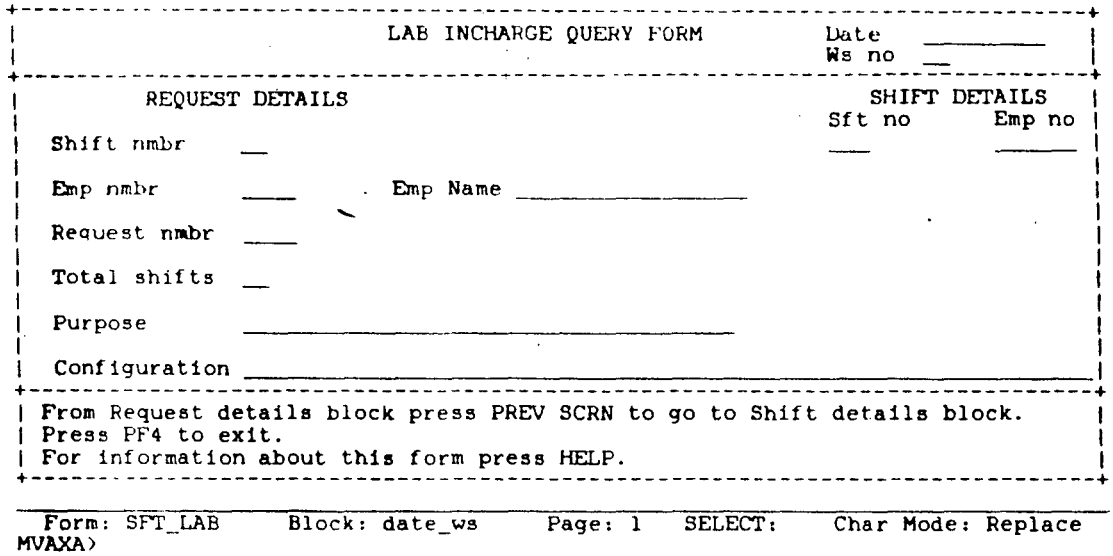

Form: SFT_MOD_CA Block: MOD_CANC Page: 1 SELECT: Char Mode: Replace
MVAXA>


PATCH_LINK
WS #
PATCH DESC
STATUS
EMP #
INST/RMVD DATE
Reason
HELP : 1. HELP to know the information about this form
2. FIND to execute a query and update the retrieved records.
3. DOWN ARROW to go to next record. 4.DO to commit. 5. PF4 to exit.
Form: SFT_PATCH_ Block: patch_link Page: 2 SELECT: Char Mode: Replace

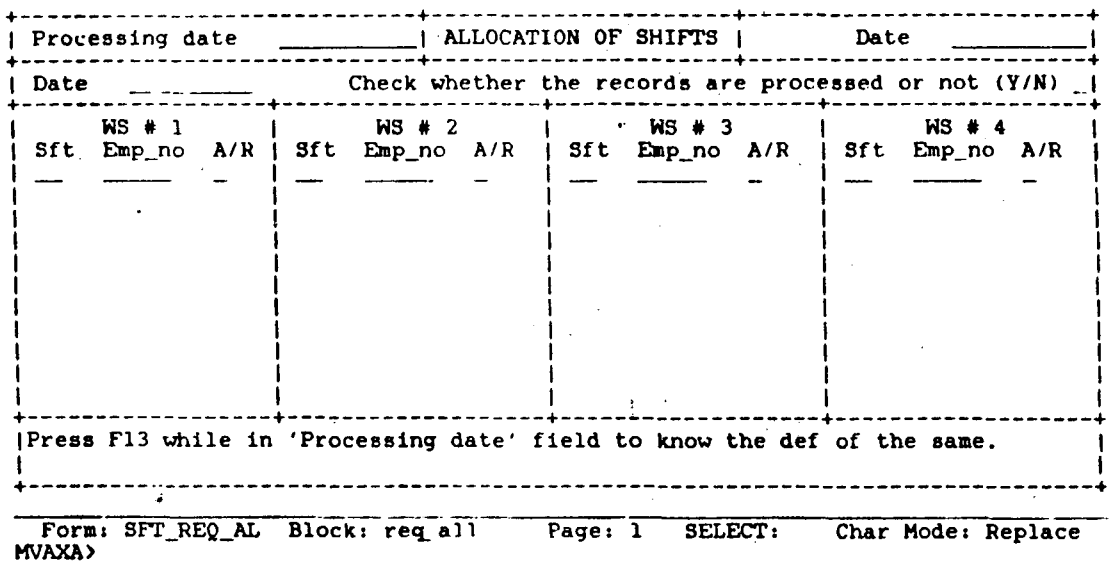

Purpose Block
1.Successful Patch Installation
2.Successful Patch Removal
3.Failure to install
4.Failure to remove
5.Query / Update 0. Exit
Enter your option and press <RETURN> : _
to continue or <EXIT> to exit.
Form: SFT_PATCH_ Block: purp Page: 1 SELECT: Char Mode: Replace
MVAXA>




Form: SFT_LOG Block: log Page: 1 SELECT: Char Mode: Replace
MVAXA>




Form: SFT_LAB Block: date_ws Page: 1 SELECT: Char Mode: Replace
MVAXA>




Form: SFT_REQ_AL Block: req_all Page: 1 SELECT: Char Mode: Replace
MVAXA>

## APPENDIX C

## COMPARITIVE STUDY OF THE DATABASE PACKAGES

## - INGRES AND ORACLE

| SUBJECT | INGRES | ORACLE |
|---|---|---|
| Database model | Relational | Relational |
| Hardware/OS on which available | VAX, mVAX etc. VMS & ULTRIX, Different m/cs UNIX , (Not available on XENIX on PC). | Same Also on XENIX on PC. |
| Whether works across network (Ethernet/DECNET) | Yes | Yes |
| Supports Transparency for location of data in a network. | Yes (INGRES/NET) | Yes (SQLNet) |

| | | |
|---|---|---|
| Gateways available for data under other DBMS | RMS Gateways for VAX/VMS & dbase III Gateway for PCs. | None. But can access data of DB II and SQL/DS |
| Form making capabilities | Default and customised form making facilities | Same |
| | - VIFRED (visual form editor )<br>- FRS (forms runtime system ). | - SQL forms. |
| Report generation capabilities. | Default & customised report generation facilities | Same |
| | - Report by form<br>- Report writer | - SQL reports<br>- Report writer |

| | | |
|---|---|---|
| Graphic report generation | Pie, bar and line charts<br><br>- VIGRAPH | Same<br><br>- SQL graph |
| PC interface | Through INGRES / PCLINK<br>PC packages like LOTUS, dbase III, wordstar can be used. | Through SQL*Calc<br>- Dbase files can be used on data conversion.<br>- has got a built-spread sheet facility<br><br>-SQL calc |
| Working in multiuser environment | Yes | Yes |

| | | |
|---|---|---|
| Query support | Has SQL support<br>- IBM DB2 compatible, Ansi standard.<br>- Has data transfer facility.<br>- Date & money data types accepted. | Has SQL support<br>- Same<br>- Same<br>- Same |
| Data handling capability | - any number of databases, tables per db, rows in a table<br>- max. 127 fields per record.<br>- max. 2000 chars per record. | - Same<br>- max. 255 fields per record.<br>- max. 128K chars per record.<br>- max. 64K chars for a field. |

| | | |
|---|---|---|
| ɔcking facilities | Read & Write locks at<br>- database level<br>- table level<br>- page of a table level | Shared update locks at<br>- database level<br>- table level<br>- row ( record level ) |
| Locking facilities | Various options available | Same |
| Deadlock Resolution | Automatic Deadlock detection and required rollback is done | Same |