292

DESIGN AND IMPLEMENTATION OF A LIBRARY MANAGEMENT SYSTEM

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

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE

625

by

NAGARAJ GOVINDARAJ

SCHOOL OF COMPUTER AND SYSTEM SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI - 110 067

DECEMBER 1989

CERTIFICATE

This work titled **DESIGN AND IMPLEMENTATION OF A LIBRARY MANAGEMENT SYSTEM** has been carried out by **Mr. Nagaraj Govindaraj**, a bonafide student of the School of Computer and System Sciences, Jawaharlal Nehru University.

This work is original and has not been submitted so far in part or full for any degree or diploma in any other University or Institute.

Nagaraj Govindaraj
Candidate

[illegible]
Supervisor
SCSS, JNU
18/12/89

[illegible]
Dean
SCSS, JNU

ACKNOWLEDGEMENTS

I thank my guide **Dr. S.Bhalla** for first teaching me about DBMS and later on guiding me on this project.

I thank the **Dean** and **faculty** of the School of Computer and System Sciences at J.N.U. for teaching me Computer Science in a friendly and lucid fashion.

I thank **Mr. M.N. Diwakar** and **Ms. H. Uma** of HCL Ltd. for their guidance and encouragement and all the facilities given to me at HCL.

I thank my parents for their support which has encouraged me to do my M.Tech. and my wife for being patient during these five months and for helping me with preparing this report.

Nagaraj Govindaraj

Nagaraj Govindaraj

December 1989

PREFACE

My M.Tech project represents the culmination of a long standing interest in Database Management Systems. I have taken up this project because I felt that a working knowledge of applications development around a database management system (DBMS) would give a good exposure to all facets of a modern DBMS which would be of immense help in my proposed career in systems software I have devoted my time this last semester of my M.Tech. to two areas.

1. developing a working application (Library Management System)
2. exploring the features of the DBMS used (ORACLE)

ORACLE gives excellent tools for developing applications. Its rapid prototyping features has permitted me to complete a fairly large sized application within the stipulated period of four months.

I am grateful to my guide, Dr. S.Bhalla for focussing my interest in this fascinating area and to HCL Ltd. for giving me an interesting project to work in.

Nagaraj Govindaraj

December 1989

ABSTRACT

A feature analysis of ORACLE is presented first, based on guidelines set in the working paper of the Relational Database Task Group of ANSI/X3/SPARC "Feature Catalogue of Relational Concepts, Languages and Systems". Studying the features of ORACLE gives an idea of the tools available for applications development. An indepth study has revealed some drawbacks which are laid out.

Before looking at the specific application, the organisation of a generalised Library Management System is evolved and the requirements of automation for library management are presented.

Finally the work done in designing and implementing a working library management system is covered. The requirements of the system are drawn up, the policy decisions taken are highlighted and the features of the implementation are presented. A sample form layout is described to illustrate the design of a form based application.

CONTENTS

PAGE NO.

PREFACE
ABSTRACT

1. ORACLE AS A DATABASE 1

1.1 Introduction 1

1.2 Architecture of the ORACLE relational database management system 2

1.3 Salient features of ORACLE 2

1.3.1 Background 2
1.3.2 Relational Characteristics 3
1.3.3 Logical Database Description 3
1.3.4 Database Storage 5
1.3.5 Functional Capabilities 6
1.3.6 Distributed Data Capabilities 8
1.3.7 Interface Descriptions 9
1.3.8 Administrative Tools 13
1.3.9 Operational Aspects 14

1.4 Drawbacks of ORACLE 20

2. AN OVERVIEW OF LIBRARY MANAGEMENT SYSTEMS 23

2.1 Introduction 23

2.2 An History of Library Automation 24

2.3 Organisation of a Generalised Library Management System 25

2.4 Requirements of a Computerised Library Management System 29

3. DESIGN AND IMPLEMENTATION OF A LIBRARY MANAGEMENT SYSTEM 31

3.1 Introduction to the Application 32

3.2 Requirements of a Computerised System 32

3.3 Policy Decisions 35

3.4 Components of the Library Management System 38

3.5 Description of a Sample Layout 43

3.6 Implementation Details 46

APPENDICES 50

REFERENCES 55

CHAPTER 1
ORACLE AS A DATABASE

1.1 INTRODUCTION

Recent times have seen a qualitative shift in software technology which has brought software tools that both ease and speed up applications development. ORACLE is one such tool that has grown out of the need to make stored information available in a form compatible to the user's need and to redefine applications easily in a short time in response to users needs.

ORACLE is a comprehensive platform for developing applications around a SQL-based relational database management system. It is currently offered on a wide range of computers ranging from large mainframes to personal computers, while maintaining the same interface across the entire range and thus providing portability across many hardware environments. It includes several powerful application development and generation tools that provide complete facilities to system designers for designing, developing and testing software products built around the ORACLE relational database management system.

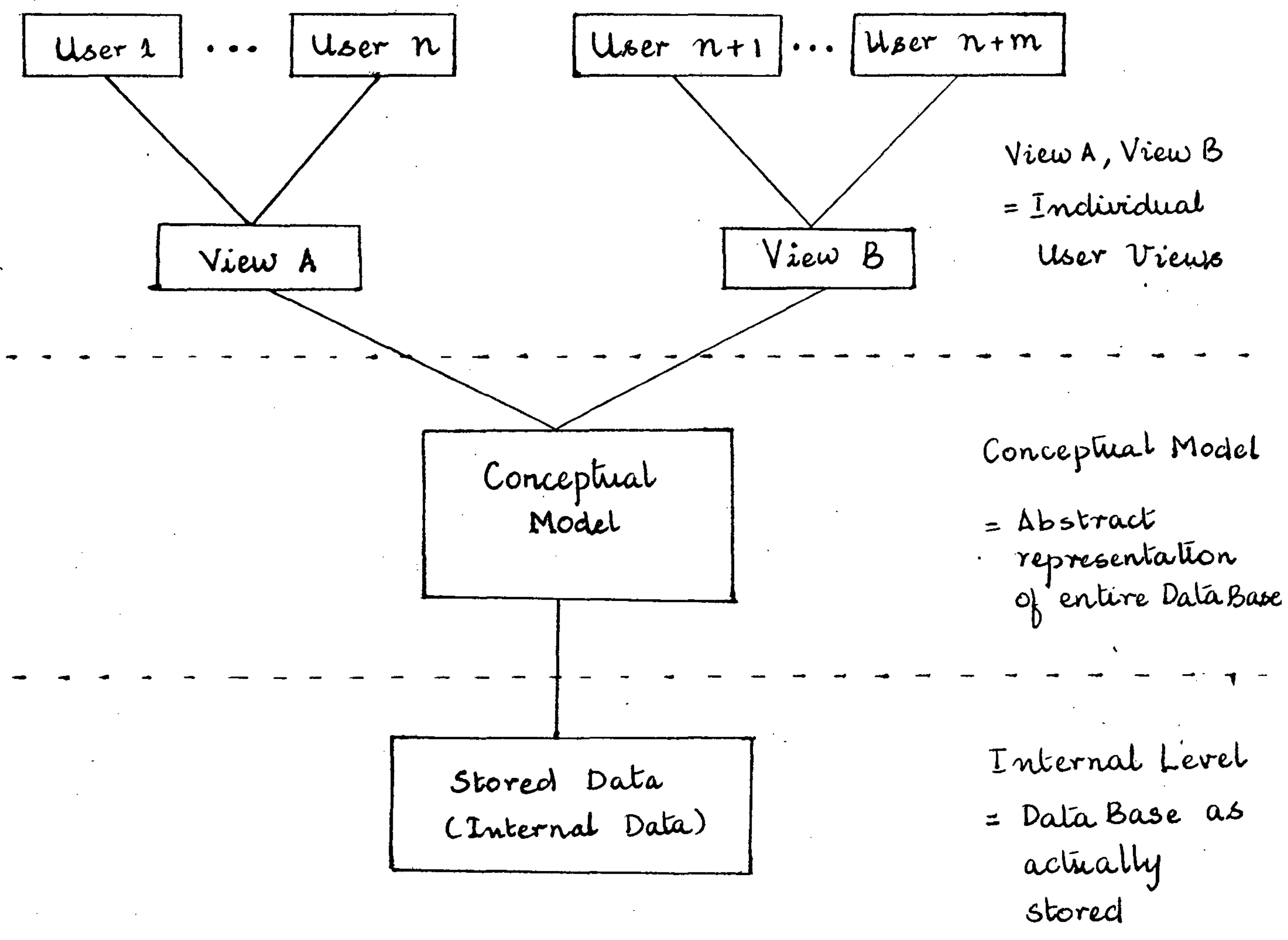
1.2 ARCHITECTURE OF THE ORACLE RELATIONAL DATABASE MANAGEMENT SYSTEM

The architecture relates the various languages or interfaces of the ORACLE Relational Database Management System [SCHM83]. In the architecture presented, we are interested only in those interfaces or languages used by humans and we are not concerned with interfaces between system modules and other implementation details. The external interfaces resemble that of SYSTEM R, the prototype Relational Database Management System built by IBM (the internal structures are different however).

1.3 SALIENT FEATURES OF ORACLE

1.3.1 Background

ORACLE is intended to be a system for naive users and experienced developers requiring a flexible Database Management System. It is especially suited for applications that require granularity of security or that may be run on different hardware. In its current state, it supports distributed data management also. Its compatibility to both IBM and ANSI versions of SQL (refer appendix A) along with its numerous tools and utilities, make it an excellent choice for applications developers.



User 1
...
User n
User n+1
...
User n+m
View A
View B
View A, View B = Individual User Views
Conceptual Model
Conceptual Model = Abstract representation of entire DataBase
Stored Data (Internal Data)
Internal Level = Data Base as actually stored

FIG. 1 AN ARCHITECTURE OF ORACLE

1.3.2 Relational Characteristics

E.F. Codd has established twelve rules that a system would have to support to be considered **fully relational** (refer Appendix B). ORACLE offers full adherence to eight of these twelve rules and partial conformance to the remaining four. The four rules which are partially supported are :

Rule 2 :	A table definition in ORACLE does not specify a PRIMARY KEY for that table.
Rule 5 :	Integrity constraints cannot be specified through SQL in ORACLE.
Rule 7 :	Insert, update or delete of rows of a view derived from more than one table are not allowed even where they are theoretically possible.
Rule 10 :	Integrity constraints are not stored in the catalog.

1.3.3 Logical Database Description

An ORACLE database consists of

1. a set of table definitions which are stored in system maintained tables. (Data Dictionary)
2. a set of stored rows for each table
3. a set of views.

A database is named by assigning the desired name to the file when the database is created.

Tables within a database are assigned unique names at the time of definition. Duplicate rows are permitted in a table. As ORACLE does not support the definition of keys for a row, uniqueness of rows may be enforced only by creation of a unique index that enforces uniqueness of a row having a certain value for the indexed columns.

Columns within a row are assigned unique names when the table is defined. They can be aliased in SQL commands referencing them and can be manipulated in selection and retrieval operations using the relational comparison operators, arithmetic operators and aggregate functions. Columns are assigned data types at the time of table definition. ORACLE supports the following datatypes :

1. CHAR (variable length alphanumeric string)
2. NUMBER (floating point decimal)
3. DATE (there is only one storage format that includes date and time, but several possible output formats are supported in SQL).

4. LONG (variable length alphanumeric strings containing upto 65536 characters)
5. RAW (byte-oriented data uninterpreted by ORACLE)
6. LONG .RAW (uninterpreted byte oriented data upto 65535 bytes long)

User defined data types (DOMAINS) are not supported.

1.3.4 Database storage

An ORACLE database consists of a database file corresponding to one or more partitions. A newly installed database always has a SYSTEM partition which contains

1. the Data Dictionary tables
2. temporary tables
3. online help tables

Additional partitions may be added at any time for controlling placement of data across disks.

Partitions are logical units, each of which corresponds to atleast one physical file. Files are added to a partition as required and may reside anywhere on the operating system. A single partition may contain many database tables;

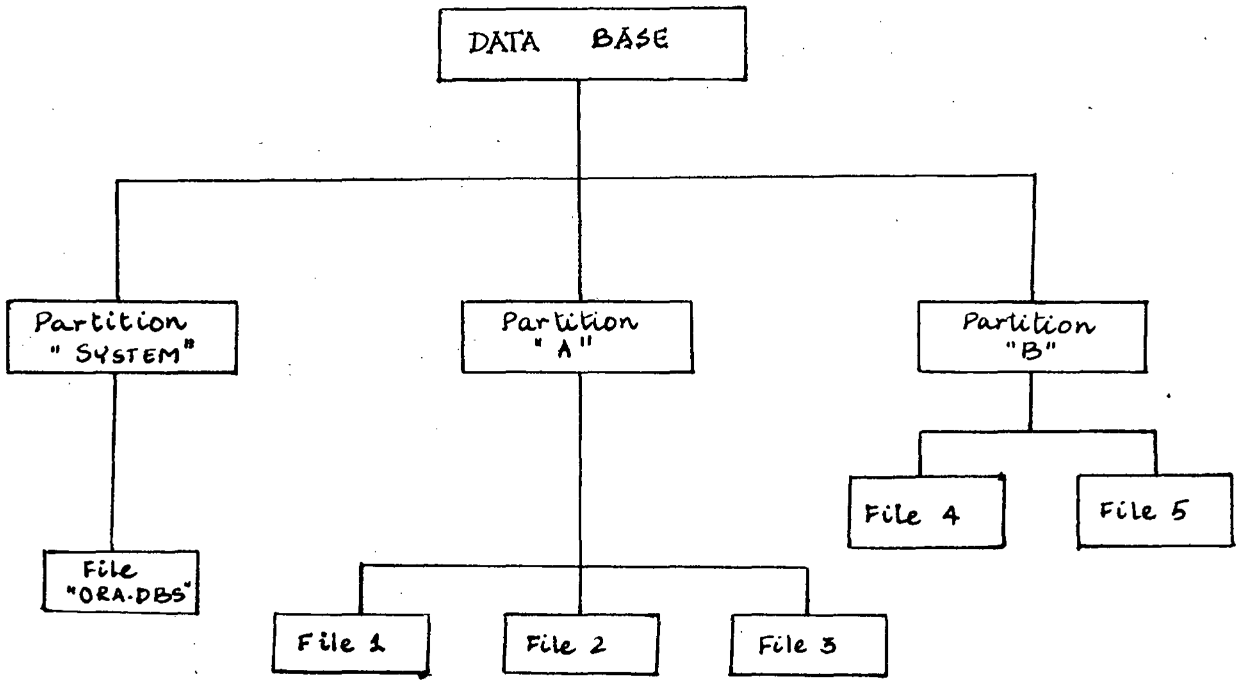


FIG.2 PHYSICAL STRUCTURE OF AN ORACLE DATABASE

a table may span over all or several files of a partition, but may not span partitions. Indexes must be stored in the same parition as the tables which they index.

Space is allocated for a table and possible indexes at table creation time. How space is allocated to the table and its indexes is also defined during creation time. The maximum possible size of a table can be set to exceed any possible physical limits.

1.3.5 Functional Capabilities

The functional capabilities are those used to select and manipulate database constituents [SCHM83]. All the functional capabilities of ORACLE are provided through the SQL language in the form of the SQL*PLUS and embedded SQL interfaces. The functions provided through SQL can be discussed under 3 categories.

a. The Data Definition Language (DDL) : Users are allowed to use CREATE, ALTER and DROP for creating new objects, altering the structure of existing objects and removing objects from the system.

b. The Data Manipulation Language (DML). This is used to query the database or to change the stored data. A versaitile SELECT statement with several options is provided which serves to select rows for answering queries. It is also used in subqueries to select rows for modification. UPDATE, INSERT and DELETE are used to alter existing rows, insert new rows or remove one or more rows from a table.

c. The Data Control Language (DCL) : This is used to control access to the database and to commit or abort transactions. A GRANT statement with several options is used for granting logon privileges to the ORACLE system, controlling access to tables and to confer various rights to users on various database tables. COMMIT and ROLL BACK allow a transaction to be made permanent or be nullified.

In addition, ORACLE provides handling of **null** values for a field of any type. Nulls are appropriate when the actual value is unknown or when a value would not be meaningful. Nulls may be manipulated in DML statements by use of the NVL function for eg. NVL (COM, 0) returns 0 when COM is null and the value of COM if it is not null.

1.3.6 Distributed Data Capabilities

ORACLE offers the following distributed data capabilities as additional functional capabilities in its latest version. [DOWG89].

a. **Location transparency** of tables is provided to users (but not to the DBA). Each machine operating in a distributed mode has information in its catalogue concerning the location and method of communication for reaching remote tables. This ensures autonomy of each site in a distributed database.

b. **Distributed Queries** are handled as follows :

   1. The local machine looks up the address of all remote tables.

   2. It performs some optimisation on the original query and then formulates sub queries against the remote tables which are then sent to the remote locations.

   3. The remote node receiving these sub-queries optimises them just as if they were generated locally and then sends the results back across the network.

c. **Distributed Update** is possible in ORACLE wherein a single transaction can update tables in two or more machines. The **two-phase commit** protocol is used to implement distributed updates. This protocol consists of participating locations to first agree to prepare to commit and then agreeing that the transaction was successfully completed before actually committing.

d. **Transparent table migration** is also handled wherein a table can be moved from one machine to another without requiring the DBA to update the catalogues on each machine.

1.3.7 Inferface Descriptions

Here we describe the interfaces used by users in interacting with ORACLE. [FAIR88, PERR89, WOJT89].

1. SQL*PLUS

This is the SQL-based interactive interface offered to the user. Its primary purpose is the creation and manipulation of tables and views within the database. It provides a full implementation of ANSI standard SQL with extensions to allow report formatting, saving and retrieving commands and execution of host operating system commands. An important feature is the extensive on-line help provided.

2. SQL*FORMS

This is an approximation to a Fourth Generation Language allowing non-procedural design of applications through creation of **forms.** Forms are composed of smaller logical units termed as **blocks.** A block may include data gathered from several tables, compute new fields for display and if required update a single table. Since a complex application may use several tables, one block is defined for each table being updated. Forms can call other forms and the user can move from screen to screen as a result of single or multiple actions or conditions.

The program execution logic of a form-based application is controlled implicitly by means of **triggers.** Triggers consist of command procedures that are executed when the user performs an action or some event takes place. These procedures are formed from SQL commands and other types of instructions (for e.g. assignment and string manipulation operations). Typically a trigger is attached to the form to edit the data inter-relationship between blocks before committing the transaction to the database. This will ensure that the database consistency is maintained. Triggers may also be used to validate data entry, to retrieve data for display and to execute user exits (procedures written in a high-level language to carry out tasks not possible in SQL).

3. SQL*Reports

This is an SQL-based report definition and generation language. SQL is used to define and process the data to be extracted from the database. Further facilities are provided to determine the display of the retrieved data.

Report processing is carried out in 2 steps

a. A report generator (RPT) reads the report specification producing an intermediate file that combine extracted data with the formatting instructions. The formating commands are similar to those used by nroff, the UNIX formatter.

b. A report formatter (RPF) reads the intermediate file and produces the report.

SQL*Reports is limited in that it cannot handle heavily formatted reports.

4. Embedded SQL

ORACLE offers a convenient way to incorporate SQL statements in programs written in high-level programming languages. Such programs are run through a precompiler to convert SQL statements into corresponding statements in the programming language. The output of the precompiler is compiled with the host language's compiler. Such precompilers are provided for COBOL, C, FORTRAN, ADA and PASCAL.

5. Other Interfaces

They include

a. **SQL*Menu** : A sophisticated non procedural tool for integrating different ORACLE functions and forms into a menu driven application.

b. **SQL*Calc** : offers a spreadsheet environment as a user interface

c. **SQL*Graph** : Translates database queries into equivalent graphic representations.

d. **Easy*SQL** : Offers beginners a simple way of building queries and reports without learning SQL.

e. **SQL*QMX** : A report writer where queries are entered by example.

f. **SQL*NET** : A product to access remote ORACLE databases.

g. **SQL*Connect** : A product to access IBM's mainframe relational databases such as DB2 and SQL/DS from ORACLE applications.

1.3.8 Administrative Tools

Administrative tools are essential to manage the database effectively [PERR89]. They are used for

a. monitoring details of storage structure.
b. backing up and restoring the database.
c. monitoring database performance.
d. controlling and monitoring user access to the database.

They include

1. **The Data Dictionary** : This forms the central documentation system of ORACLE and is essential for managing the database. It stores user names, user access rights, table names and column names, table storage information and auditing data for disaster recovery. This data is stored in the form of tables. The data dictionary is automatically

updated whenever changes are carried out. Even ordinary users can use it to gain information regarding their own activities.

2. **IOR** : It is used for starting, stopping and initialising the system.
3. **SGI** : It provides information regarding global memory usage by the system.
4. **ODS** : Monitors and displays system use for active users.
5. **AIJ** : Provides after-image journalling when enabled.
6. **CRT** : Used for making known to ORACLE screen display characteristics of all terminals on the machine running ORACLE.
7. **EXP** : Exports data from the database to a system file which can then be used for archiving or moving data between operating systems or ORACLE databases.
8. **IMP** : Imports data from a file to the database
9. **ODL** : Used for loading data from files foreign to ORACLE.
10. **CCF** : Creates a database file.

1.3.9 Operational Aspects

1. Optimising System Performance

Indexing and **Clustering** of tables may be used to optimise database performance [PERR89].

a. **Indexing**

Oracle automatically maintains indexes and uses them to speed up queries when query optimisation is being done. Indexes are useful in two ways.

i. To minimise database access time by significantly reducing the number of disk input and output operations as frequently queries can be answered by examining the indexes rather than the database.

ii. To ensure that table rows are uniquely identifiable by the column(s) upon which the index is based.

b. **Clustering**

The performance of frequently used joins may be improved by clustering together the joined tables. Clustering is done over the columns used in the join (clustered columns).

The effects of clustering a group of tables are :

i. The rows from all the tables that have the same value in the clustered columns are stored in the same disk page.

ii. Each distinct value in a clustered column is stored only once in the database.

iii. A **cluster index** is created on the clustered columns. This will improve the performance of operations that require searching on the clustered columns.

2. Security

ORACLE maintains tight security over all aspects of the database [PERR89]. Access to the database is maintained by an authentication system that requires a user name and password for every authorised user.

Two levels of privileges are maintained by ORACLE :

a. **System privileges** determine who can logon to ORACLE and whether or not a user can authorise other users to logon to ORACLE

b. **Object privileges** provide detailed control on database objects. The user owning an object (base table or view) can grant or revoke any or all of the following privileges to other users : read, insert, update, alter definition, create indexes and grant privileges to other users. The person defining the object (the owner) has all privileges over the object.

3. Auditing

Auditing is used to monitor and record security-relevant events. ORACLE provides a number of auditing commands and associated tables for storing auditing information. Auditing has to be enabled before auditing data can be collected and can be done at two levels.

a. **System level** : System wide activity such as logon, data definition and passing on privileges is recorded.

b. **Table level** : Owners of objects can record all accesses to objects they own.

4. Concurrency Control

Locks are used in ORACLE to prevent inconsistent situations that may arise due to multiple users concurrently accessing and modifying the database. Locking in ORACLE is fully automatic and does not require user intervention. However users can override the default locking through certain SQL statements. The types of locks available in ORACLE to users are :

a. **Share Locks** : They permit users to query data, but not change it. Use of SHARE locks is appropriate when a user wants to query the table only and wants to ensure that other users do not change the table between queries. Several users may place SHARE locks at the same time on a table.

b. **Share Update locks** : They permit several users to simultaneously both query and update a table. A user can select several rows of a table for locking in SHARE UPDATE mode. At the same time, other users can query the table or lock other rows in SHARE UPDATE mode. Thus concurrency is improved through use of this type of lock.

c. **Exclusive locks** : An EXCLUSIVE lock permits other users to query data, but not change it, Other users are not allowed to place any lock on the same data. It is used

    i. in programs to prevent deadlock by ensuring that each user will lock certain tables in the same order.

    ii. when extensive changes are to be made to a table and other users have to be prevented from updating the table.

In addition, ORACLE has DDL (Data Dictionary Language) locks to lock the data dictionary so that changes to database structures do not interfere with ongoing user operations.

All established locks endure until COMMIT is done. Then all established locks are released.

ORACLE detects deadlocks automatically. Deadlock resolution is done by terminating one of the processes causing the deadlock.

5. Crash Recovery

Roll-backward recovery is managed automatically by the system through the provision of **Before-image journalling** [FAIR88]. The before-image of each record being updated in a database is preserved. If a failure occurs in mid-transaction, the entire transaction is backed out and the database is restored to its original state from the before-image file. This protects against most system failures with the exception of device failures such as head crash on a disk drive.

Roll-forward recovery is provided through **After-image journalling** [FAIR88]. When enabled, an after-image of every update transaction is preserved once it has been successfully applied to the database. The after-image is stored on a separate device from the database and provides a log of all activity performed on the database since the last backup. It will protect against disk drive failure as the after-image can be applied to the corresponding backup and make the database current.

A backup of the database is done by shutting down the system and copying all database files to the backup device.

1.4 DRAWBACKS OF ORACLE

ORACLE was found to be a flexible easy-to-use system with most of the features expected from a modern full-fledged database management system [SCHM83, SHAW88]. However a few drawbacks were noticed with the version of ORACLE used (version 5.1).

1. The only **integrity constraints** allowed in ORACLE are
   a. preventing two rows in a table from having the same value in a specified column through definition of UNIQUE indexes.
   b. preventing a column having a null value (NOT NULL).

   The inability to define other integrity constraints in the data dictionary means a lot of code checking for constraint violations has to be duplicated between applications using the same data.

2. User-defined **domains** are not supported. Provision of domains is required to ensure consistency of data across columns that store the same or related information. This is useful for supporting primary and foreign keys.

3. Views derived from two or more tables do not support insert, update, delete even where theoretically possible.

4. Stored procedures are not supported. Stored procedures are collections of precompiled SQL statements, the use of which will speed up operations since code need not be generated.

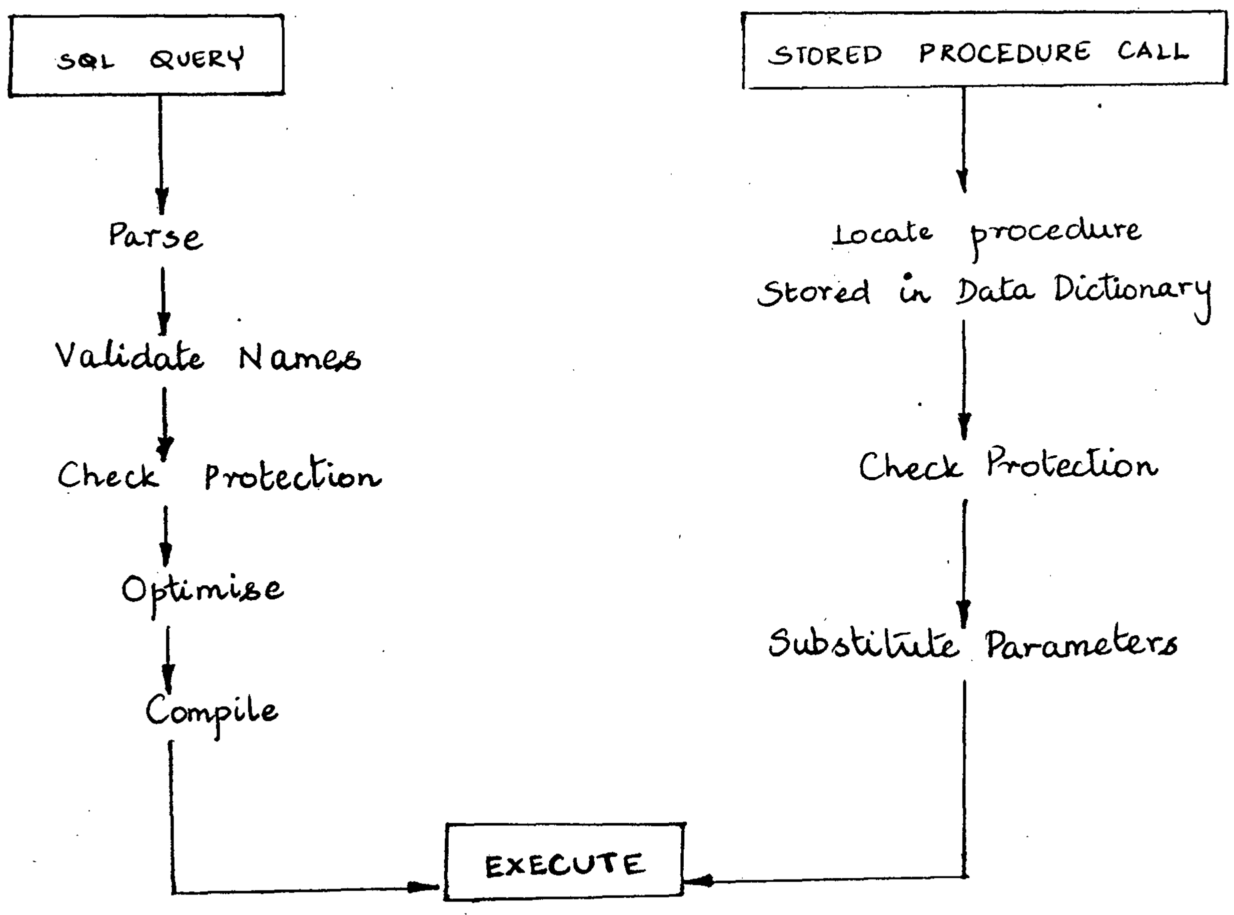
5. The system does not maintain statistical information regarding the distribution of data which can be used for query optimisation.

6. Online backups are not supported. The database has to be shut down before a backup or recovery can be done.

7. Parameters that determine characteristics of ORACLE's operation may not be changed when the system is up. The system must be shut down, the settings of the parameters changed and the system restarted for the new parameters to take effect.

8. After Image Journalling in ORACLE keeps a record of each block written to the database. Even if only a byte is updated, the whole block gets written out. Thus the After Image file grows rapidly.

9. While ORACLE was found suitable for the usual decision support applications, high volume transaction support is lacking. Decision support applications typically handle a small number of users retrieving data with adhoc queries. Online applications are far more demanding, supporting dozens to hundreds of users who are concurrently updating large databases. Transactions are expected to execute giving real time response in such online applications.



A Stored Procedure is processed in less time than the corresponding Sequence of SQL Commands

FIG.3 HOW STORED PROCEDURES IMPROVE PERFORMANCE

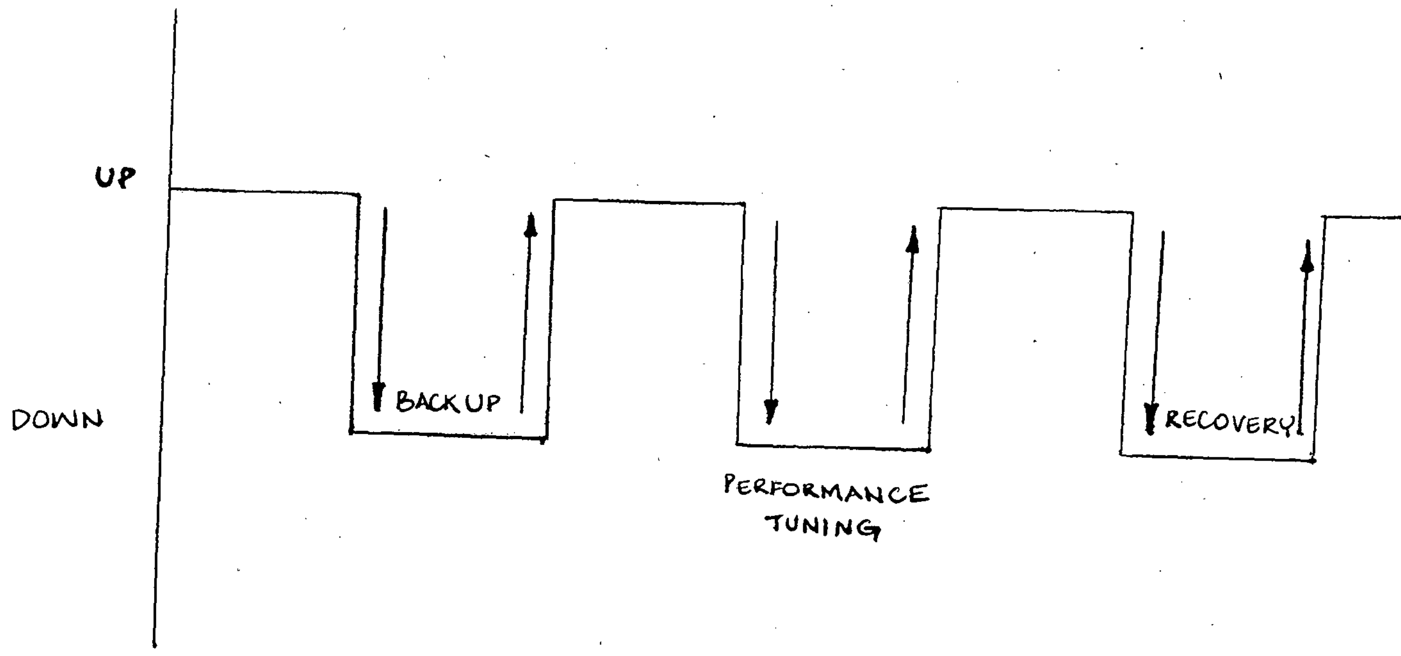


FIG.4 INTERMITTENT AVAILABILITY OF ORACLE

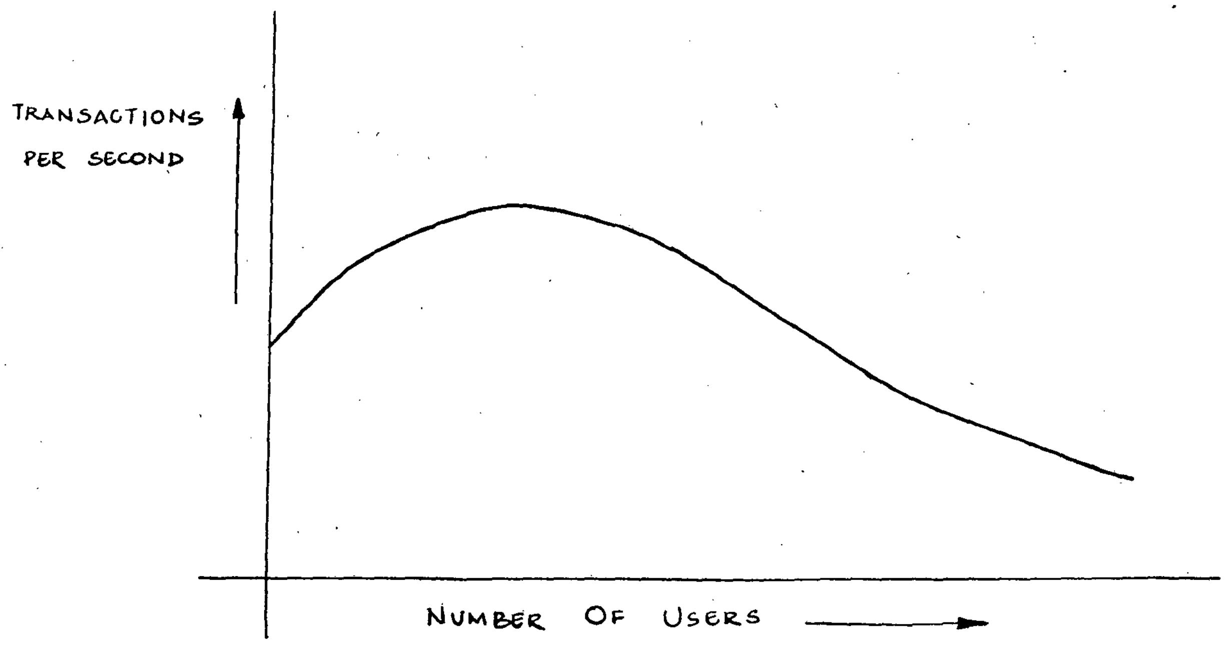


FIG. 5 LOW ONLINE THROUGHPUT OF ORACLE

CHAPTER 2

AN OVERVIEW OF LIBRARY MANAGEMENT SYSTEMS

2.1 INTRODUCTION

The advent of the Information Age has caused a tremendous growth in size and services offered by libraries. It has become imperative for libraries to automate their procedures because of the following drawbacks with the manual systems.

1. There is no way in which a single item can be located in a library of thousands of volumes in a matter of minutes.

2. It is frequently the case that outdated information is maintained in the manual catalogue as updates are done only at selected periods, say once a month. Also catalogue cards may get damaged due to frequent usage.

3. It is not always possible for libraries to employ more people when they increase in size. The increased work load thus poses a burden on the existing staff.

2.2 AN HISTORY OF LIBRARY AUTOMATION

The climb towards automation of libraries has occured in three stages [FAYE83].

In the first stage, in the early 1960s, computers were used as high speed typewriters and storage devices to produce catalogue cards and to prepare book and microform catalogues. This enables these labor intensive tasks to be done faster and at a lower cost.

In the second stage, in the early 1970s, database vendors offered shared cataloguing services online from a central database. This permitted libraries to share the cost of cataloguing their materials. Such online catalogues have helped ease subsequent efforts in automating libraries.

The third stage started when it was realised that many of the tedious record keeping functions in the daily operation of a library could be automated such as circulations, acquisitions and losses. Such automation would contain costs of expansion and provide valuable information to decision makers. In addition, users of libraries can quickly locate information from the online database.

The future may see the advent of the **"paperless library"** where all literature is available online to the user. Current advances in secondary storage technology such as optical disks and CD ROMS make one feel that the day is not far off when it will be economical to keep all information online.

2.3 ORGANISATION OF A GENERALISED LIBRARY MANAGEMENT SYSTEM

An integrated library system usually supports several functions [FAYE83] such as

(1) Reference services
(2) Acquisitions
(3) Cataloguing
(4) Serials receipt
(5) Inter library loan
(6) Circulation

A library has to examine its need for automating any of these functions before deciding to do so. These functions are discussed below in more detail.

Reference Services

A major difficulty is conventional card catalogues is in providing enought access points so that any required text is located based on the user's current need. This means that the search should ideally not require a pre-knowledge of the catalogue organisation. Also such catalogues are available only at the library. They may become inconsistent with time due to changes in the cataloguing system such as change in practice or change in the rules.

The following reasons call for automation of the catalogue

(a) Access to large collections will probably be difficult without an online catalogue.

(b) Remote access can be provided through dial-up if required.

(c) The online data may be required for automating other functions such as circulation.

Acquisitions

Acquisition services deal with adding items to the library collection. This service should maintain knowledge of pending orders, vendors with whom they are placed, purchase costs, reminders to be sent to vendors etc. Much of the work is highly repetitive and hence acquisition is a suitable function for automation.

Cataloguing

This is the process of creating the online catalogue. It is usually the first step in automation as the resulting online data can be used to automate other functions. Once a cataloguing method has been chosen, automating the cataloguing function should be easy given the powers of the computer.

Serials receipt

It is necessary to keep track of subscriptions to periodicals and whether all issues to-date have been received. Automation is desirable since manual systems lack a suitable way to alert the library when an item has not been delivered. An additional function of automated serials receipts is to indicate when a complete volume is ready for binding.

A major difficulty in automation lies in the exceptions and irregularities of the procedure. An example is the wide range of frequencies of publication (from daily to yearly). Therefore this function remains one of the least likely candidates for automation.

Interlibrary loans

Sometimes libraries may loan to other libraries some of their own items. This is made easy by keeping a central online catalogue of several libraries. Whenever a library discovers an item is not in its stock, it can locate the item with another library through the central catalogue and forward a request for that item to the other library.

This function is a good candidate for automation as it facilitates

1. Location of an item with another library.
2. Keeping track of loaned items.

Circulation

Automating the circulation frees the library staff of many tedious book-keeping tasks. It also provides the management information regarding users' interests for guiding acquisitions and regarding usage of items in the library.

Conclusion

It is necessary to isolate those tasks in library management that can be done only by the library staff and pick up the best candidates for automation out of the above functions. This is possible only through careful determination of requirements and listing out of specifications before designing a system for automating library management.

2.4 REQUIREMENTS OF A COMPUTERISED LIBRARY MANAGEMENT SYSTEM

1. The primary requirement is access to a Computer. Such access may be got by
   a. buying a Computer.
   b. getting access to a Computer where Computer time is loaned out.
   c. purchasing a turnkey system from a vendor. This is a good choice if the library has no systems personnel among its employees.

2. The data regarding the library must be available online with a database management system to manage it and interface to other applications. Communications software may be essential to handle communication with remote terminals and computers.

3. Applications software is needed to actually perform the library functions in coordination with the database management system. Software must be designed and implemented keeping in view the local requirements.

4. Terminals must be provided wherever people need to access the library's database. Graphics terminals may be used to give a good user interface. Touch panels may be used where people are nervous of using computers. All the types of terminals used must be capable of interfacing to the applications software.

5. Computers fail occasionally and storage devices get damaged. Backup of data must be taken or a backup hardware configuration may be provided.

6. Orientation of staff is necessary to overcome their fears and apprehensions. Comprehensive training should be provided to the staff so that they will be confident in using the system. A good user interface can build confidence in a short time.

CHAPTER 3

DESIGN AND IMPLEMENTATION OF A LIBRARY MANAGEMENT SYSTEM

3.1 INTRODUCTION TO THE APPLICATION

HCL has excellent library facilities at its R & D Centre at Madras. The library has the latest books, manuals and journals in computers and related areas. A librarian takes care of all library related activites. His functions are to take care of

a. Maintenance of the library catalogue.

b. Answering queries regarding the stock.

c. Circulation of library stock.

d. Keeping track of acquisitions.

e. Reports for management.

Since in-house computer facilities are available with advanced packages like ORACLE, it was felt that computerisation of library management could be done at little extra cost other than the time spent by an engineer. A few advantages of computerisation are :

a. The librarian could be freed from tedious book keeping tasks.

b. Instant access to the online catalogue with powerful query handling capabilities is assured to the users.

c. Complex reports can be generated for management in a very short time.

Computerisation of library management was therefore assigned to me as a project from August - December 1989.

3.2 REQUIREMENTS OF A COMPUTERISED SYSTEM

The requirements were drawn up taking into consideration three view points.

a. Users' requirements

b. Librarian's requirements

c. Management's requirements.

Users' Requirements

i. The users should be informed of the latest additions to the library.

ii. Users need to consult the catalogue often. Provision of an easy to use catalogue with enough access points will help reduce the tedium of searching.

iii. Users need to be provided with reservation services so they can have guaranteed access to popular literature in a certain time. They should be informed when a reserved book is returned.

iv. Users need to be reminded of their overdue borrowings so that they return them in time.

Librarian's Requirements

i. The librarian should be spared the tedium of routine tasks such as cataloguing, circulation of items and searching the catalogue on a user's request.

ii. Reservation services should be handled independently by users without intervention of the librarian.

iii. Reminders to users for overdue items should be generated automatically.

iv. Reports for management should be easy to generate.

Management's requirements

i. Information regarding the library stock such as the catalogue, lists of latest additions should be easily accessible to all employees so that the best usage can be made of it.

ii. Circulation should be well managed so that all users can be assured access to an item in a certain time ; importantly it should known who has an item so that a person with urgent need can get it from him/her.

iii. Usage statistics of journals need to be maintained so that infrequently used journals need not be subscribed for.

iv. Subscription details to journals need to be maintained so that reminders regarding non-receipt of journal issues can be sent to the agents.

v. Printouts of the catalogue need to be generated so that verification of library stock can be carried out.

vi. Information regarding items issued to an employee and dues of an employee such as items lost need to be maintained so that it is made sure that all dues are cleared at the time of leaving.

3.3 POLICY DECISIONS

Regarding Implementation

A working system with full features was to be implemented within the allotted period of four months by one person. ORACLE was chosen as the medium of implementation because

a. ORACLE is a full featured modern data base management system and all the needs of the application can be met through it.

b. In view of the limited manpower and time available (four man months), the rapid prototyping capabilities of ORACLE was a great advantage.

c. ORACLE was available in-house.

Regarding Functioning of the Library

The following decisions were taken regarding the general functioning of the library.

a. The items in stock (books, journals, manuals and backvolumes) will be issued to members on production of an appropriate card.

b. Details of members and cards issued to them should be registered in the library database. The mail addresses of members on the computer are stored for sending messages to them. The number of cards issued to a member is variable.

c. Books/Manuals are classified into three categories :

* Ordinary
* In-demand
* Reference

An **ordinary** book may be issued for a period of ten days

An **in-demand** book is issued for a period of three days only

An **reference** book is not issued at all.

The status of a book may be changed depending on demand for the book.

d. Journals are classified into 3 types :

* regular issues
* special issues
* proceedings of societies such as ACM.

Old journals are to be bound into back-volumes after a certain time.

e. The library offers the following services :

1. **Catalogue querying on the following entry points :**

   i. for books : Title, Author, Subject, Publication-year, Publisher.

   ii. for manuals : Name, Source, Publication-Year.

   iii. for journals : Name, journal type, Issue-date.

   Details may be entered with wild cards.

2. **Reservation Service** : Reservations are allowed for books and manuals. Members are intimated through electronic mail when the reservations become available and should collect the item within two days or forfeit the reservation. A maximum of nine reservations per item is allowed.

3. **Cancellation Services** : Members can cancel their reservation on an item.

4. **Reminder Services** : A member who has not returned an item will be informed through electronic mail once the item is overdue.

5. **Information regarding latest additions** : Lists of latest additions of journals, books and manuals will be displayed on the notice board from time to time.

6. **Fines** are collected for overdue items. The rate depends on the item and is in direct proportion to the number of days overdue.

Regarding Maintenance of the Libary

The library database should mirror the current status of the library correctly. This can be done if

i. All new items are catalogued as soon as they arrive.

ii. All new changes regarding status of items such as books being sent for binding are logged immediately.

iii. Reminders to agents for journals are to be generated when issues are not received.

iv. Exhaustive catalogue printing is to be done to keep a record of the items on hand whenever necessary.

3.4 COMPONENTS OF THE LIBRARY MANAGEMENT SYSTEM

The library system is organised into two major components as follows :

* Librarian's Utility
* User's Utility

These utilities were developed using the SQL*Forms package.

1. Librarian's utility

This is the interface of the librarian to the Library Management System. It is a menu-driven system where choosing a menu option either yields another menu or at the lowest level, a form for carrying out some operation.

The librarian's functions can be divided into five main classes.

1. Library-maintenance related
2. Circulation-related
3. Queries against the catalogue
4. Generating reports
5. Backups

These are the options provided in the main menu. In addition, if any reminders need to be sent for overdue items, the librarian is reminded through a status field. He need only press a key for the reminders to be mailed to the members.

Maintenance Functions

These are

a. Addition of new items
b. Addition of a copy of an existing item
c. Modification of details of existing items

and can be for any of the following item classes.

a. Books

b. Manuals

c. Journals

and d. Back volumes

In addition, subscription details may be added or modified and memberships may be added, modified on deleted.

Circulation Functions

These are

a. Issues

b. Extensions

c. Returns

d. Registering losses

and can be for

a. Books

b. Manuals

c. Journals

and d. Back volumes

Due date calculation is done automatically during issues and extensions, and takes care of holidays also. During Returns, fines are calculated and displayed based on the fine-rate and days overdue.

Queries against the catalogue

This function serves to trace items in the online catalogue based on certain entry points.

For example, books can be searched for based on any combination of the following entry points.

a. Catalogue id of the book
b. Title
c. Author
d. Publisher
and e. Year of publication.

All details of items qualifying the search criteria are retrieved and displayed.

Report Generation

The following reports can be generated for books, manuals and journals

a. Full catalogue
b. Latest additions

Backups

All the tables in the library database are backed up using the EXPORT utility of ORACLE.

2. <u>User's Utility</u>

This is the interface provided to the users and is available to all users on a login account accessible to everyone on the computer system. The functions provided to the user can be categorised into three classes.

a. Reservations and cancellations
b. Queries against the catalogue
c. Information regarding new additions.

Reservations

The item being reserved and the reserver are identified through a form. The reservation is assigned a priority and added to a table maintaining reservations.

Cancellations

The item on which reservation is to be cancelled and the reserver are identified. Details of the reservation will be deleted from the table maintaining reservations.

Catalogue Queries

The same function provided in the librarian's utility is provided here.

Latest Additions

The list of latest additions for any of the following items is retrieved and displayed.

a. Books

b. Manuals

c. Journals

and d. Backvolumes

3.5 DESCRIPTION OF A SAMPLE LAYOUT

Catalogue Queries for books is chosen to illustrate the typical implementation of a function through forms.

Tables used

Details of books are stored in three tables :

i. TITLE table gives the details of each book in the library.

ii. BOOK table gives the details of each copy of all the books.

iii. CARD table gives details of all books issued from the library.

The details of table are shown in Figure 7.

Form Layout

The layout is shown in Figure 8.

The form is divided into three blocks

i. The CONTROL block in which the criteria for search are established through filling up the values for any or all of the fields displayed (wild cards are also allowed).

ii. The TITLE block in which the details of each book satisfying the search criteria are shown.

iii. The COPY block in which the details of each copy of the book are shown.

Form Operation

i. The user enters values for any or all of the fields in the CONTROL block using CARRIAGE-RETURN key to move between fields. He presses the KEY-EXEQRY function key to start the search.

ii. Records satisfying the search criteria are retrieved into the other two blocks. The TITLE block displays one record at a time from the list of selected records from the TITLE table. The COPY block displays one record at a time from the list of selected records from BOOK table corresponding to the currently displayed record in the TITLE block.

iii. a. The user uses KEY-NXTREC and KEY-PRVREC, function keys to move between the records in the TITLE block. Each time a new record is fetched into the TITLE block, the corresponding list of COPY records is fetched into the COPY block.

b. The user uses CARRIAGE-RETURN to move to the COPY block to examine the details of copies. He uses KEY-NXTREC, KEY-PRVREC function keys to move between the list of COPY records.

c. He presses the KEY-NXTBLK function key to move back into the CONTROL block if he wants to refine the search criteria by either modifying the values entered or entering fresh values for fields. He presses KEY-EXEQRY to start the search again.

d. He uses the KEY-EXIT function key to stop the search at any time and exit the form.

3.6 IMPLEMENTATION DETAILS

i. Detailed design of each function was done by first giving details of
   a. form layout
   b. form operation by user
   c. database operations done through form processing logic.

ii. Maintenance related functions were implemented first. This was to make the librarian familiar with the forms interface and create an online uptodate database at the earliest.

iii. Catalogue Querying was provided next so that users could have access to the online catalogue.

iv. Report generation was provided through SQL*Reports.

v. Management of circulation was then provided

vi. All the functions were integrated into a menu-driven system to give the Librarian's utility first and then the User's utility.

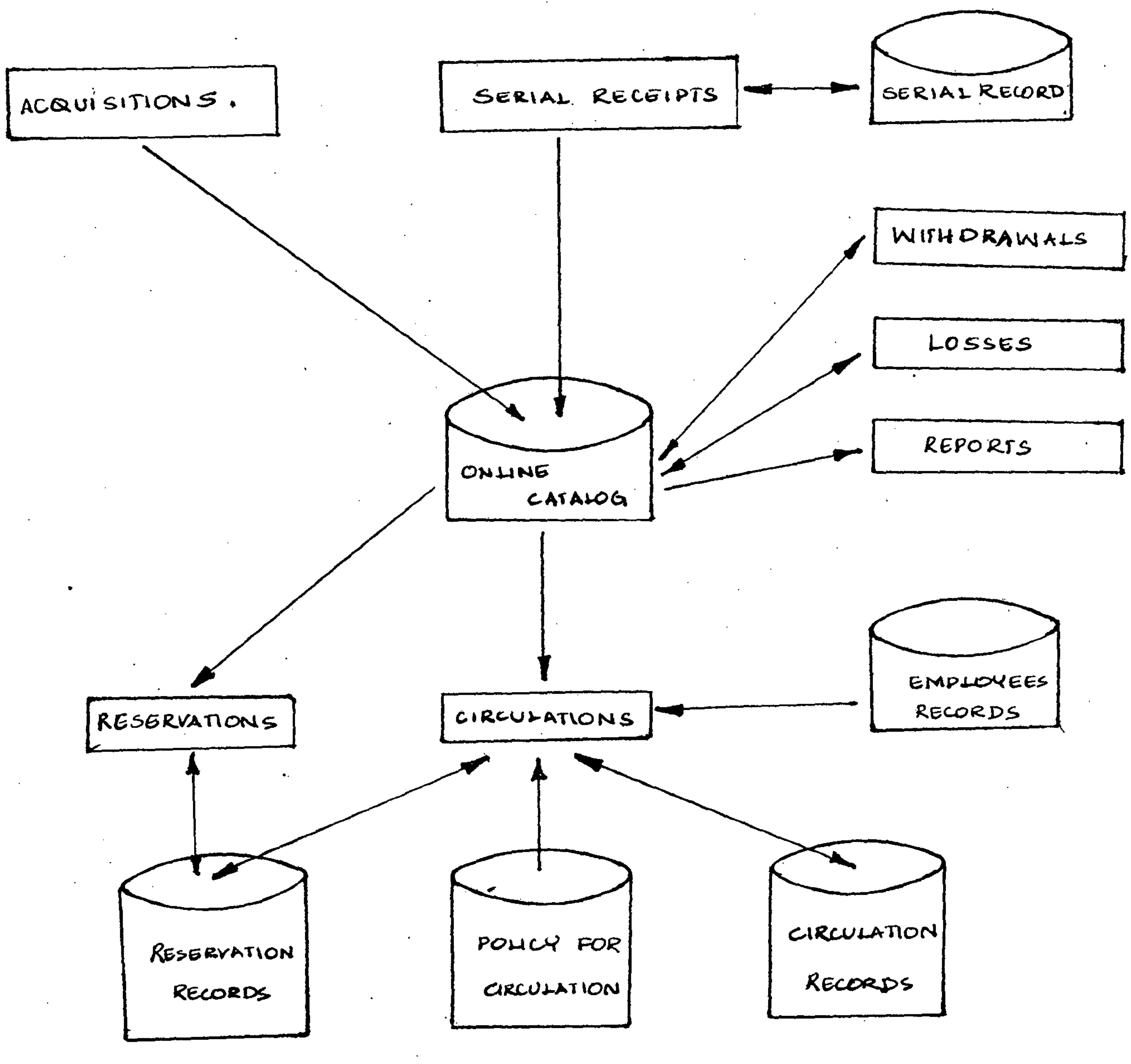


FIG. 6 MODEL OF THE LIBRARY MANAGEMENT SYSTEM

FIG. 7 TABLES USED IN SAMPLE APPLICATION (1)

1. **TITLE Table (gives details of each book in the library)**

   Key is TITLE-ID

No.	Field	Field Type	Explanation
1.	Title-id	CHAR(5)	Unique identifier for a title in the collection of books
2.	Subject1	CHAR (15)	Upto 3 subjects with
3.	Subject2	CHAR (15)	which the book
4.	Subject3	CHAR (15)	is concerned about
5.	Title	CHAR (70)	Name of book
6.	Author	CHAR (50)	
7.	Total-copies	NUMBER (2)	Total number of copies
8.	Avail-copies	NUMBER (2)	Copies available now in library
9.	Reservations	NUMBER (2)	Size of reservation queue on this title

FIG. 7 TABLES USED IN SAMPLE APPLICATION (2)

2. Book table (given details of each copy of all books)

Key is TITLE-ID + COPY-ID.

No.	Field	Field-type	Explanation
1.	Title-id	CHAR (5)	Uniquely identifies a book
2.	Copy-id	NUMBER (2)	with its title-id and copy-id
3.	Publisher	CHAR (5)	Publisher's Name
4.	Public-Year	NUMBER (4)	Year of Publication
5.	Cost	NUMBER (6,2)	Book-value in currency
6.	Currency	CHAR (1)	
7.	Acq-date	DATE	When the book was bought
8.	Avail	CHAR (1)	Is it in the library or not ?
9.	Status	CHAR (1)	Ordinary, In demand or Reference
10.	Remarks	CHAR (25)	

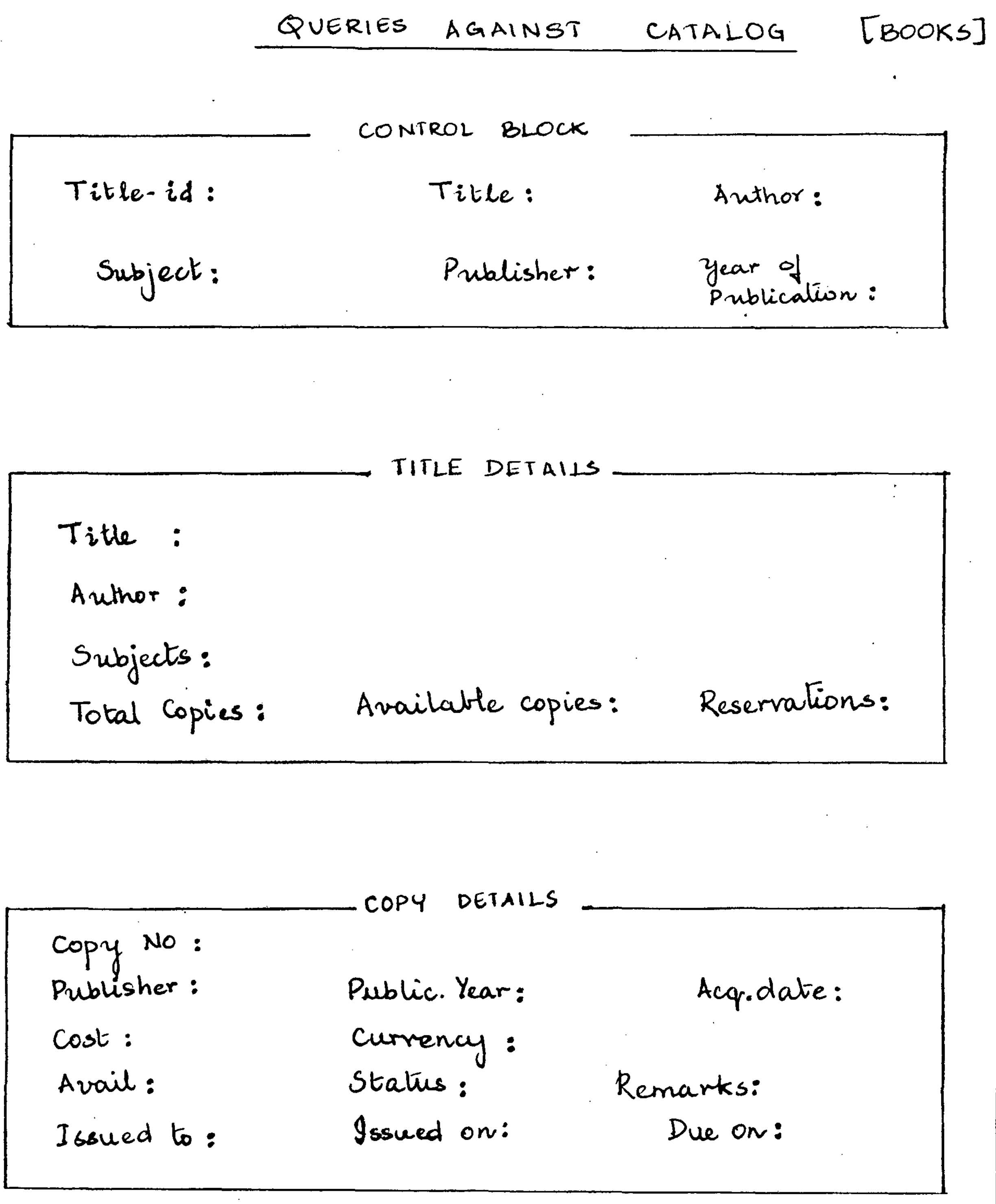
FIG.7 TABLES USED IN SAMPLE APPLICATION (3)

3. **CARD Table (gives details of books issued from library)**

Key is TITLE-ID + COPY-ID

No.	Field	Field Type	Explanation
1.	Title-id	CHAR (5)	Identifies copy of
2.	Copy-id	NUMBER (2)	book issued
3.	Emp-no	" "	Identifies employee and the
4.	Card-no	" "	card of employee used to borrow the book
5.	Issued-on	DATE	
6.	Due-on	DATE	
7.	No.-remin	NUMBER (2)	How many reminders were sent to return the book
8.	Last-remin	DATE	When the last reminder was sent

NOTE : Unique indexes are used to ensure unqiueness of primary key values in the tables shown.



QUERIES AGAINST CATALOG [BOOKS]

CONTROL BLOCK

Title-id :	Title :	Author :
Subject :	Publisher :	Year of Publication :

TITLE DETAILS

Title :

Author :

Subjects :

Total Copies :	Available copies :	Reservations :

COPY DETAILS

Copy No :

Publisher :	Public. Year :	Acq. date :
Cost :	Currency :	
Avail :	Status :	Remarks :
Issued to :	Issued on :	Due on :

FIG. 8 SAMPLE FORM LAYOUT

APPENDIX A

SQL

SEQUEL, later called as SQL, was one of the earliest relational languages based on the Relational Model of data [SHAW88]. It was developed by D.D. Chambertin and his associates at IBM's San Jose Research Laboratory in 1974. In 1977, SYSTEM R, a relational database prototype based on SQL became operational at IBM, San Jose. Its success led IBM to develop a number of relational products around SQL. The success of IBM's relational products such as DB2 (introduced in 1983) prompted other vendors to offer their own SQL-based products. In the past, applications developers have been sceptical of the ability of relational DBMSS to perform as well as traditional hierarchical and network models. The success of SQL-based relational database management systems has made these experts question their scepticism [ROTH89].

SQL offers a full implementation of the relational algebra, a set-oriented language for manipulation and retrieval of data stored as per the relational model. It can be used stand alone or embedded in application programs for database manipulation. It has no inherent program control statements or operations. Usage of SQL in embedded form eliminates the need for an application development language to know anything about the structure or nature of the database being manipulated.

In 1986, SQL was ratified by ANSI as the standard relational database language. This together with support from IBM has established SQL as the defacto standard for Relational Database Management Systems.

APPENDIX B

CODD'S TWELVE RULES

E.F. Codd in his paper "A Relational Model of Data for Large Shared Data Banks" (Comm. ACM, June 1970) [CODD70] established the principles of the relational model. His additional refinements to the relational model resulted in "12 rules for deciding how relational a DBMS Product is " [SHAW88]. These twelve rules are used for testing how well a database management system fits the relational model. They are laid down below:

Rule 1 : All information in a relational database is represented explicity at the logical level and in exactly one way - by values in R-tables.

Rule 2 : Each and every datum (atomic value) in a relational database is guaranteed to be logically accessible by resorting to a combination of R-table name, primary key value and column name.

Rule 3 : Indicators (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing at the logical level the fact that the information is missing in a systematic way-independent of data type. Besides the logical representation, the DBMS must support manipulative functions for these indicators and these must also be independent of the data type of the missing information.

Rule 4 : The database description is represented at the logical level just like ordinary data, so that authorised users can apply the same relational language to its interrogation as they apply to the regular data.

Rule 5 : A relational DBMS must support at least one language

(1) whose statements are expressible as per some well-defined syntax as character strings

and

(2) which offers comprehensive support for **all** the following items.

(a) Data definition (b) view definition

(c) Data manipulation (interactive and through program)

(d) Integrity constraints (e) Authorisation

(f) transaction boundaries (begin, commit, roll-back).

Rule 6 : The DBMS includes an algorithm at least as powerful as VU-1 for determining (at view definition time) whether that view is tuple-insertable and tuple-deletable and whether each of its columns is updatable. It records this info in the catalog.

Rule	7	:	The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also for the insertion, updation and deletion of data.
Rule	8	:	Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods.
Rule	9	:	Application programs and terminal activities remain logically unimpaired when information - preserving changes of any kind that theoretically permit unimpairment are made to the base tables.
Rule	10	:	Integrity constraints specific to a particular relational database must be specified in the relational data sub-language and storable in the catalog (not in the application programs).
Rule	11	:	A relational DBMS has distribution independence.
Rule	12	:	If a relational system has a low level (single-record-at-a time) language, that level cannot be used to subvert or bypass the integrity rules and constraints expressed in the high level relational language (multiple-records-at-a-time).

REFERENCES

[CODD70] Codd, E.F. "A Relational Model for Large Shared Data Banks". CACM 13, 6 Jun. 1970.

[CODD82] Codd, E.F. "Relational Database : A Practical Foundation for Productivity". CACM 25, 2, Feb. 1982.

[DATE81] Date, C.J. "An Introduction to Database Sytems (3rd ed.)". Narosa Publishing House, 1981.

[DOWG89] Dowgiallo, E. "The Art of Distribution". DBMS, Mar. 1989, pp.42-55.

[FAIR88] Fair, P. "Testing a Trio of 4 GLS". Unix Review, Sep. 1988, pp. 83-89.

[FAYE83] Fayen, E.G. "The Online Catalog : Improving Public Access to Library Materials". Knowledge Industry Publications Inc., New York, 1983.

[FINK88] Finkelstein, R. and Pascal, F. "SQL Database Management Systems", BYTE, Jan. 1988.

[LIAR89] Liardet, M. and McCrone, A. "Benchtest of ORACLE". Business Computer, Mar. 1989, pp. 45-54.

[PERR89] Perry, J. and Lateer, J.G. "Understanding ORACLE". BPB Publications, 1989.

[ROTH89] Roth, D. "Striving for Optimisation". Oracle Magazine, Volume III/Number 1, Spring 1989.

[SCHM83] Schmidt, J.W. and Brodie, M.L. "Relational Database Systems - Analysis and Comparison". Springer-Verlag, 1983.

[SHAW88] Shaw, R.H. "SQL : An Emerging Database Standard for PCs". PC Magazine, May 17, 1988.

[WOJT89] Wojtkowski, W.G. and Wojtkowski, W. "Applications software programming with Fourth-Generation Languages". Boyd & Fraser Publishing Company, Boston, 1989.