1. *Chlorophyll a* (Chl *a*)

90% $\pm$ 1% IN METHYLENE BLUE TESTING SINCE 1974

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

**"A TOOLSET FOR DATABASE ADMINISTRATION
IN A MULTIPLE DATABASE ENVIRONMENT"**

by

DENNIS Tat Ming CHEUNG, B.Sc.

**A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Science**

**Faculty of Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
December 1991**

**@ copyright
1991, Dennis Tat Ming Cheung**

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-76011-7

Canada

The undersigned hereby recommend to the
Faculty of Graduate Studies and Research
acceptance of the thesis,

**"A TOOLSET FOR DATABASE ADMINISTRATION
IN A MULTIPLE DATABASE ENVIRONMENT"**

submitted by Dennis Tat Ming CHEUNG, B.Sc.,
in partial fulfilment of the requirements
for the degree of Master of Science

Thesis Supervisor

Chairman,
Department of Systems and
Computer Engineering

Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University
December 1991

ABSTRACT

This thesis describes the functions performed by a Database Administrator (DBA) in a Relational Database Management System (RDBMS) environment and identifies those that are feasible for automation. Problems with database administration in a multiple database environment are identified and an automated software tool is presented to assist database administration in such an environment.

An architecture for the toolset is proposed and its components are identified. The proposed system is called Multiple Database Administration System (MDAS) and permits the DBA to manage different RDBMS databases of the enterprise running on various hardware platforms. The data architecture of MDAS is based on the Information Resource Dictionary System (IRDS) standard proposed by American National Standard Institute (ANSI).

A proof of concept prototype for the toolset has been designed and developed. The approaches introduced in this thesis are currently being applied in database administration of multiple RDBMS databases of a large organization in Canada.

ACKNOWLEDGEMENTS

I would like to express my thanks to my thesis supervisor, Dr. Morteza Niktash for his continuous support, guidance and encouragement during the exploration of this very exciting field. His patience and willingness to discuss the numerous technical alternatives were instrumental in helping to coalesce the sparks of ideas into this thesis.

I am also greatly indebted to Michael Fleming and Rainer Kossmann for discussing some of the concepts contained in this thesis and for reviewing the contents of this thesis.

The greatest thanks of all goes to my wife Ivy, who spent many late nights to type this thesis and stood by me throughout the years of studying, never questioned my commitment and gave me the strong spiritual support which I needed to reach my goal.

Finally I would also like to thank my 2 boys, Kenny(7) and Jamie(5) for giving dad a break by not running around me while I was writing up my thesis.

TABLE OF CONTENTS

	Page
Acceptance Sheet	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Figure	viii
List of Tables	x
Glossary of Acronyms	xi
Glossary of Terms	xii
CHAPTER 1: INTRODUCTION	1-1
1.1 Motivation	1-4
1.2 Thesis Objectives	1-5
1.3 Thesis Overview	1-7
CHAPTER 2: OVERVIEW OF DATABASE ADMINISTRATION FUNCTIONS	2-1
2.1 Survey of Database Administration Functions	2-6
2.1.1 Environment Creation and Maintenance	2-8
2.1.2 DBMS Configuration Control and Maintenance	2-11
2.1.3 Security Administration	2-14
2.1.4 Operation Monitoring & Control	2-16
2.1.5 Space Usage and Monitoring	2-20
2.1.6 System Resource Usage Monitoring, Performance Tuning	2-24
2.1.7 Application Development Review and Evaluation	2-30
2.1.8 Dictionary Administration	2-36
2.1.9 Other DBA Administrative Activities	2-38
2.2 Survey of Database Administration Tools	2-39
2.3 Summary	2-42

	Page
CHAPTER 3: DATABASE ADMINISTRATION IN A MULTIPLE DATABASE ENVIRONMENT	3-1
3.1 Overview	3-3
3.2 Issues in Database Administration	3-15
3.2.1 Focus of Current DBA Tools	3-16
3.2.2 Proliferation of Databases	3-19
3.2.3 Multiple Sites	3-22
3.2.4 Heterogeneous Environment	3-24
3.2.5 Data Complexity	3-26
3.2.6 Growth and Changing Operating Environment	3-28
3.3 Problem Definition	3-29
3.3.1 Problems with Managing Multiple Database	3-30
3.3.2 Problems facing the DBAs	3-33
3.3.3 Example	3-36
3.4 A Proposed Approach to Manage Multiple Databases	3-42
3.4.1 A Data Dictionary for Multiple Databases	3-44
3.4.2 DBA Toolset for Multiple Databases	3-47
CHAPTER 4: A PROPOSED DBA TOOLSET ARCHITECTURE	4-1
4.1 The IRDS Standard and Its Role	4-4
4.1.1 ANSI IRDS - Overview	4-8
4.1.2 ISO IRDS - Overview	4-12
4.1.3 Discussion	4-15
4.2 DBA Toolset Architecture - Overview	4-17
4.2.1 Data Architecture	4-19
4.2.2 Software Architecture	4-23
4.3 DBA Toolset Components - Design Description	4-28
4.3.1 The Information Resource Dictionary	4-30
4.3.1.1 IRD Schema Layer	4-34
4.3.1.2 IRD Data Layer	4-37
4.3.2 The MDAS Interface	4-41
4.3.3 The Catalog .xtraction Programs	4-44
4.3.4 The Access Programs	4-50
4.3.5 The Command Programs	4-53
4.3.6 The IRD Operation Programs	4-57
4.3.7 MDAS Administration Programs	4-59

	Page
CHAPTER 5: DESIGN AND DEVELOPMENT OF A DBA TOOLSET PROTOTYPE	5-1
5.1 Functions Automated by the Prototype	5-8
5.2 Toolset Components	5-13
5.2.1 The Information Resource Dictionary	5-13
5.2.1.1 IRD Data Layer	5-15
5.2.1.2 IRD Schema Layer	5-16
5.2.2 The MDAS Interface	5-28
5.2.3 The Catalog Extraction Programs	5-32
5.2.4 The Access Programs	5-37
5.2.5 The Command Programs	5-40
5.2.6 The IRD Operation Programs	5-44
5.2.7 MDAS Administration Programs	5-47
5.3 Populating the DBA Toolset IRD	5-49
CHAPTER 6: THESIS SUMMARY	6-1
6.1 Contributions	6-2
6.2 Limitations and Constraints	6-6
6.3 Conclusions	6-10
6.4 Future Development	6-12
REFERENCES	R-1
APPENDIX A -- Survey of DBA Tools	A-1
APPENDIX B -- Overview of ANSI IRDS	B-1
APPENDIX C -- Overview of ISO IRDS	C-1
APPENDIX D -- Detailed Design for IRD Layers	D-1
APPENDIX E -- Sample Programs, Screens and Output Listings	E-1
E1 -- MDAS Interface Program	E1-1
E2 -- Catalog Extraction Programs	E2-1
E3 -- Access Programs	E3-1
E4 -- Command Programs	E4-1
E5 -- IRD Operations Programs	E5-1
E6 -- MDAS Administration Programs	E6-1

LIST OF FIGURES

FIGURE	DESCRIPTION	PAGE
2.1	Overview of a DBMS Architecture	2-3
2.2	Grouping of Major Database Administration Functions	2-7
2.3	Layers of Software	2-12
3.1	Centralized Database Environment	3-4
3.2	Client-Server Centralized Database Environment	3-5
3.3	Distributed Database Environment	3-8
3.4	Multiple Database Environment	3-12
3.5	Focus of Current DBA Tools	3-18
3.6	Proliferation of Databases	3-21
3.7	Multiple Sites	3-23
3.8	Heterogeneous Environment	3-25
3.9	Data Complexity	3-27
3.10	Typical Activities for DBA for Managing Multiple Databases	3-32
3.11	Existing Database Environment at AECL Research	3-37
3.12	Multiple Database Environment at Chalk River Laboratories	3-39
3.13	Data Flow among System Catalogs and IRD	3-46
4.1	Overview of ANSI IRDS Architecture	4-10
4.1A	Overview of ISO IRDS Architecture	4-13
4.2	Data Architecture for the DBA Toolset	4-22
4.3	MDAS Architecture	4-25
4.4	Types of Information at the IRD Schema Layer	4-36
4.5	Types of Information at the IRD Data Layer	4-38
4.6	Examples of MDAS Interface	4-43
4.7	The Concept of Catalog Extraction Process	4-49
4.8	Sample Access Program	4-52
4.9	The Concept of Sending Commands to Target Databases	4-56

FIGURE	DESCRIPTION	PAGE
5.1	The Prototype Environment	5-5
5.2	Typical Activities for the DBA at Chalk River Laboratories	5-10
5.3	Relationships between Entities for the Prototype's IRD Data Layer	5-22
5.4	Sample Relational Tables required for an initial Toolset Prototype	5-27
5.5	Sample Main Menu and Submenus	5-31
5.6	Catalog Extraction Program - Overview	5-36
5.7	TABLESPACE Screen	5-39
5.8	Enroll User - Command Program	5-42
5.9	Available Options for Prototype's IRD Operation	5-45
B-1	The Four-Level Architecture of the IRDS	B-8
B-2	Subset of an Information Resource Dictionary	B-11
D-1	Entity-Relationship Model for Prototype's IRD Data Layer	D-6
D-2	Entity-Relationship Model for IRD Schema Layer	D-20

LIST OF TABLES

TABLE	DESCRIPTION	PAGE
4.1	ANSI Level 3 Starter Set	4-34
5.1	Type of System Catalog Information	5-11
5.1A	Information to be Captured/Command Sent at/to each Database	5-11
5.2	Entity Groupings of IRD Data Layer	5-18
5.3	Relationships Between Entities for IRD Data Layer	5-23

Glossary of Acronyms

ANSI	American National Standard Institute
AP	Access Program
ASCII	American Standard Code for Information Interchange
CASE	Computer Aided Software Engineering
CEP	Catalog Extraction Program
CP	Command Program
CPU	Central Processing Unit
DBA	Database Administrator
DBMS	Database Management System
E-R	Entity-Relationship
I/O	Input / Output
IRD	Information Resource Dictionary
IRDS	Information Resource Dictionary System
ISO	International Standard Organizations
MDAS	Multiple Database Administration System
PC	Personal Computer
RDBMS	Relational Database Management System
SQL	Structured Query Language

Glossary of Terms

Access Control The ability to manage which users or groups of users may have the privilege to retrieve, create, update or delete data that is held in a database. Access may be granted or revoked by the owner of the data. [BARK 89]

Application System A name given to an arbitrary collection of business functions, entities, programs and tables. [BARK 89]

Attribute Any detail that serves to qualify, identify, classify, quantify or express the state of an entity. Or, any description of a thing of significance. Note that each entity occurrence may only have one value of any attribute at one time. [BARK 89]

Attribute-group In an IRD at metadata level, an ordered set of two or more attributes used together (e.g., the attribute-group of DATE-TIME is made up of the attribute DATE and TIME). [LAW 88a]

Attribute-group-type In an IRD at schema level, an ordered set of two or more attributes used together (e.g., the attribute-group-type DATE-TIME-ADDED is made up of attribute-types SYSTEM-TIME and SYSTEM-DATE). [LAW 88a]

Attribute-type In an IRD at schema level, the label for a set of attributes which may be common to an entity-type or relationship-type. [LAW 88a]

Column A means of implementing an item of data within a table. It can be in character, date or number format, and be optional or mandatory. [BARK 89]

Data Administration The responsibility for definition, organization, supervision, and protection of data within an enterprise or organization. [LAW 88a]

Data Administrator A person or group that ensures the utility of data used within an organization, by defining data policies and standards, planning for the efficient use of data, coordinating data

structures among organizational components, performing logical database design, and defining data security procedures. [LAW 88a]

Data Dictionary A database for holding definitions of tables, columns and views, etc. [BARK 89]

Data Dictionary System An automated system such as an IRDS that can support one or more data dictionaries. [LAW 88a]

Data Integrity In information processing, the condition in which data is accurate, current, consistent, and complete. [LAW 88a]

Database An arbitrary collection of tables or files under the control of a database management system. [BARK 89]

Database Administrator A person or group which provides technical support for one or more databases, by defining database schemas and subschemas, maintaining data integrity and concurrency, providing physical database design for performance optimization, and enforcing the policies, standards, and procedures set by the Data Administrator. [LAW 88a]

Database File An allocation of space used in a database to hold the database data.

Database Management System (DBMS) A database management system, normally encompassing computerized management facilities that are used to structure and manipulate data, and to ensure privacy, recovery and integrity in a multi-user environment. [BARK 89]

Entity A thing of significance, whether real or imagined, about which information needs to be known or held. [BARK 89]

Entity Relationship Diagram A part of the business model produced in the strategy stage of the Business System Life Cycle. The diagram pictorially represents entities, the vital business relationships between them and the attributes used to describe them. [BARK 89]

Field A means of implementing an item of data within a file. It can be in character, date, number or other format, and be optional mandatory. [BARK 89]

Index A means of accessing one or more rows in a table with particular performance characteristics, often implemented by a B-tree structure on an RDBMS. An index may quote one or more columns and be a means of enforcing uniqueness on their values. [BARK 89]

Information Resource Dictionary (IRD) A data dictionary application managed by an IRDS; a collection of entities, relationships, and attributes used by an organization to model its information environment. [LAW 88a]

Information Resource Dictionary System (IRDS) A set of standard specifications for a data dictionary system resulting from U.S. Federal and national standards efforts; a computer software system conforming to these standards that provides facilities for recording, storing, and processing descriptions of an organization's significant information and information processing resources. [LAW 88a]

Information Resource Management (IRM) The responsibility for planning, organizing, and controlling information for coordinated use in data management, data processing, data communications, and data conversion, in a manner consistent with the primary goals and objectives of the enterprise. [LAW 88a]

IRD Schema A model of the logical structure of the IRD, consisting of components such as entity-types, relationship-type, and attribute-types. [LAW 88a]

IRD Schema Extensibility The capability to add new IRD schema descriptors (i.e., new entity-types, attribute-type, and attribute-types) to any IRD schema. [LAW 88a]

Metadata Information describing the characteristics of data; data or information about data; descriptive information about an organization's data, data activities, systems, and holdings. [LAW 88a]

Program A set of computer instructions, which can enter, change or query database items, and provide many useful computer functions. [BARK 89]

Record In a non-relational database system, a record is an entry in a file, consisting of individual elements of information, which together provide full details about an aspect of the information needed by the system. The individual elements are held in fields, and all records are held in files. An example of a record might be an employee. Every detail of the employee (for example, date of birth, department code, full names) will be found in a number of fields. [BARK 89]

Relation A relation is a term that embraces the concept of both table and view. [BARK 89]

Relationship What one thing has to do with another. Or, any significant way in which two things of the same or different type may be associated. Note that it is important to name relationships. [BARK 89]

Relationship-class-type In an IRD at schema level, a set of relationship-types which use the same verb in the central position (e.g., [entity-type-1]-VERB-[entity-type-2]). [LAW 88a]

Relationship-type In an IRD at schema level, the label for a set of relationships which have similar meanings and share a set of common attribute-types. [LAW 88a]

Resource Privilege A database privilege which allows the user to create tables in a database.

Repository A collection of metadata.

Schema A collection of table definitions. [BARK 89]

SQL Structured Query Language. The internationally-accepted standard for relational systems., covering not only query but also data definition, manipulation, security and some aspects of referential integrity. [BARK 89]

Space Management Management of physical and logical space utilization in a database.

System Catalog A collection of metadata (stored in relation tables) which drives all DBMS processing activity.

Table A tabular view of data, which may be used on a relational database management system to hold one or more columns of data. It is often an implementation of an entity. Tables are the logical and perceived data structure, not the physical data structure, in a relational system. [BARK 89]

Tablespace A logical portion of a database used in allocating storage for the table data, table indexes, etc.

User In information processing, an individual, organization, or facility that makes use of an information system or other software system, such as an IRDS. [LAW 88a]

View A means of accessing a subset of the database as if it were a table. The view may:

- . be restricted to named column
- . be restricted to specific rows
- . change column names
- . derive new columns
- . give access to a combination of related tables and/or views. [BARK 89]

CHAPTER 1

INTRODUCTION

In the 90's, computing hardware is becoming a widely affordable commodity and many departments within an organization are purchasing their own hardware such as micro computers, work stations, etc. Some of these machines have databases installed. In recent years, some DataBase Management System (DBMS) vendors have given out free run-time licenses of their products on some work stations.

Moreover, most large organizations will evolve their information technology into heterogeneous environments with networks connecting large numbers of various computing devices. The number of databases on this network will increase dramatically. This will occur because databases will to be established for specialized and discrete purposes. Moreover databases will increasingly be required to support the local workstation information, and will in addition also have to integrate the local workstation information with the shared corporate information resident on network information servers. As a result of these, there are more databases within an organization, some of these are not linked due to user requirements or due to their nature (e.g. production database does not link with development database). We refer to this type of multiple separated database environment as a Multiple Database Environment.

This proliferation of databases will make database administration a formidable task. It gives rise to several problems in the effective

maintenance and support of existing databases and the addition of many new databases: performance management and database administration of growing numbers of SQL engines is increasingly labour-intensive, integrity is lost, large-scale redundancy appears, and inconsistencies flourish. There are many commercial database administration tools available to ease the increasing work load on the DataBase Administrator (DBA). However, these existing tools only handle one type of DBMS or database. Hence they are unable to deal with the multiple database environment. This calls for a new type of tool which permits the DBA to manage all databases on the enterprise network regardless of DBMS type or the type of hardware upon which they operate.

1.1 Motivation

The data dictionary has been discussed for many years as a potential solution and should be considered by database management personnel.

Data dictionary tools have been utilised in many phases of a systems development cycle: user and technical design; data design; detail design; code and test; systems test and conversion; and maintenance. Dictionary tools have also been used to manage a single database. However the data dictionary is not widely used to manage multiple databases. As the growth of organizations also reflects the growth of the database management systems, a data dictionary should be explored to assist the DBA in managing the multiple database environment

An extensible data dictionary is required in order for the DBA to capture the information on the ever changing operating environment and to provide self-descriptiveness. The ANSI Information Resource Dictionary System (IRDS) is one of the system which allows extensibility.

1.2 Thesis Objectives

Although the concepts presented in this thesis can be applied to network, hierarchical, object-oriented etc. databases, the scope of this study is confined to relational databases, since the multiple relational database (SQL engine) environment is becoming quite popular and widespread today.

One of the objective of this thesis is to examine the database administration function in a multiple database environment and to evaluate a representative set of existing relational DBA tools in order to evaluate their capabilities. Some of the shortcomings of these tools are also discussed. A problem definition produced by analysis of existing tools for management of multiple databases is developed.

This thesis proposes a solution to manage multiple databases and the proposed system is called Multiple Database Administration System (MDAS). The architecture of MDAS is a single toolset which is composed of a centralized data dictionary or repository and a set of associated tools. MDAS keeps track of the SQL engines in the network and their current characteristics. MDAS also provides the associated network tools to manage the DBA aspects of SQL engines to: extract metadata from different databases and load them into the central data dictionary; monitor the status of multiple databases; send database

commands to any databases without the need to physically logon to the remote nodes and databases; etc. The concept of the IRDS is employed in developing the data dictionary architecture and to examine its extensibility features for accommodating the database administrative functions.

The proposal is then applied to a large organization (AECL Research at Atomic Energy of Canada Limited) by developing a prototype to examine the feasibility of the DBA toolset. The prototype is itself implemented using relational technology because of its dominant and wide-spread availability, although other technology such as network or object-oriented technology could also have been used. The evaluation of the prototype in an operational environment permits the identification of limitations, constraints, and future work required for the proposed toolset.

1.3 Thesis Overview

Chapter 2 focuses on the activities of a database administrator and it also reviews and categorizes the major functions of database administration. It then examines the detailed functions in each of the categories and identifies those that are feasible for automation. Next, the importance of automation for database administration is discussed and a brief finding on the surveyed commercially available DBA tools for RDBMS systems is presented. Lastly this Chapter discusses the importance of the system catalog, the problems of managing multiple databases and the lack of DBA tools which exist to manage and administer such environments.

In Chapter 3, a brief overview of the centralized, distributed and multiple database environment is given. Then the issues of database administration in the multiple database environment are discussed. Then a problem definition relating to managing multiple databases is developed. This problem definition demonstrates the growing need for multiple database administration in a heterogeneous environment. Then the problems that are facing DBAs in such an environment are discussed. A real-world example of such an environment at AECL Research is discussed. A solution to help the DBA to manage a multiple database environment is proposed together with a toolset architecture and its

components. The toolset includes two major components: a data dictionary and a set of associated DBA Tools.

In Chapter 4, an overview of the ANSI and ISO IRDS and the roles of IRDS are discussed. Then details of the proposed toolset architecture (data dictionary and a set of DBA tool) are presented. Then, a detailed discussion of each of the components of the DBA toolset is presented.

Chapter 5 discusses the application of the MDAS concept to the real world. The prototype environment at Chalk River Laboratories of Atomic Energy of Canada Limited is described. Then the detailed design and development of each prototype toolset component is outlined and discussed.

In Chapter 6, the contributions to database administration research are discussed. Then the constraints of the prototype approach are presented. Finally, a conclusion and suggestions for future development are presented.

CHAPTER 2

OVERVIEW OF

DATABASE ADMINISTRATION FUNCTIONS

A **database** may be defined as a unified collection of data in one or more physical files controlled by a **Database Management System (DBMS)** [INMO 81]. The DBMS is the software-hardware package that makes the databases easily accessible to the users. As seen in Figure 2.1, a software portion of the DBMS that some vendors call the **DBMS server** serves as an interface between the user and the database. The DBMS server provides the software tools required to insert, update, select, delete, and load data in the database. The server also handles all I/O and memory management related to the database, and in shared systems it handles security and concurrent user problems. In short, a well-designed DBMS will provide software which makes it easy for a user to communicate with the database [JACK 88].

The database management function exists because it can contribute, uniquely, to the fulfillment of those goals by establishing and controlling a resource that provides management with information - the basis for intelligent and timely decisions. This resource, the database, has value only if it is properly managed, designed, maintained and utilized [LYON 76].

Figure 2.1 Overview of a DBMS Architecture

Moreover, significant improvement in database management is often the reason cited in justifying a database administration system. More specifically, such a system can lead to reduction in the numbers of technical staff required to maintain, operate and support the organization's databases since the frequent activities are automated and critical information is available much faster. Planning for growth and optimal database efficiency can occur on a more orderly basis if performance characteristics can be gathered over a period of time to spot trends that adversely affect database operations.

Database administration in a database management system is of great importance, and can be the determining factor of success or failure of a large DBMS. The database administration system must be integrated with the design of the DBMS, and must be flexible enough to support other database applications.

A database administration system can be considered as a set of administrative, operational, maintenance, and planning functions. Each of these functions is aimed at attaining three goals:

- database availability
- database integrity
- database performance

Administration includes a diverse range of day-to-day tasks, such as the addition of new users to the database system, security management, and management of data and applications. Operation encompasses the ongoing task of monitoring and tuning a wide range of distributed or centralized database systems through monitoring performance and making changes in operating parameters as necessary. Maintenance includes establishing a standardized database configuration, backup, recovery, and software upgrades. Planning covers the aspects of ensuring sufficient resource availability for the database system as more applications are put into production. Resource requirements are composed of disk storage, computer cycles, memory, network traffic, etc.

The task of database administration within an organization is generally assigned to the DataBase Administrator (DBA).

2.1 Survey of Database Administration Functions

The DBA function is a set of integrated activities that permit an organization to manage its data resource. The responsibilities of the DBA and their relationship to other functions of the organization vary with time and circumstance. However the general DBA functions can be grouped into the following areas and as illustrated in Figure 2.2.

- Environment Creation and Maintenance
- DBMS Configuration Control and Maintenance
- Security Administration
- Operation Monitoring and Control
- Space Usage and Monitoring
- System Resource Usage Monitoring, Performance Tuning
- Application Development Review and Evaluation
- Dictionary Administration
- Other Administration Activities

A brief description of each of these categories is presented in the following sections. A check list of the major DBA responsibilities is also presented at the end of each category. Those DBA functions that are feasible for automation are identified with [Tool].

Figure 2.2 Grouping of Major Database Administration Functions

2.1.1 Environment Creation and Maintenance

Organizations which have invested in relational database technology typically will implement multiple occurrences of the vendor's database on their computer systems. The organization may decide to have one database which is the repository for the official, or production data. The organization may have a second database which is used by systems developers for the development of new applications. A third database may be implemented solely for purposes of testing new or changed applications.

As the organization's investment in computer systems and in relational database technology grows, an increasing number of databases will be implemented. It is not uncommon for medium sized organization to have 10-20 databases in use, each serving a different community of users. With increasing use of more powerful workstations, it is also anticipated that there will be rapid growth of databases in the workstation environment to support local information management and control of the information space on each workstation.

To gain administrative control over this type of environment, the Database Administrator must establish a common, or standard DBMS environment for the organization. The standard environment addresses such issues as object naming conventions, access controls, parameter

settings, backup/recovery procedures, etc. Once the procedures and guidelines for the standard environment are in place (and adhered to), a number of benefits result:

- creation of a new database in the organization becomes a straight forward process which can be accomplished in a minimum amount of time;
- monitoring and control of multiple databases, although increasingly time-consuming, can be accomplished in a routine manner since each of the databases "looks" like the next;
- backup/recovery activities also become relatively routine matters since one backup and one recovery procedure will work for any of the organization's databases.

With a standard DBMS environment in place, the DBA can then perform the functions and activities related to the maintenance of this environment.

A check list of major DBA functions in this category is presented below:

- creates new databases in response to user needs [Tool]
- establishes and standardizes common commands to access the databases [Tool]
- standardize naming conventions for databases and their related physical and logical files
- standardizes and establishes commands and procedures for backup and recovery [Tool].

2.1.2 DBMS Configuration Control and Maintenance

Database maintenance is one of the important tasks that keeps the database in good shape. There is always a recurring need for the DBA to perform regular maintenance of the database. However the DBA must carefully plan the maintenance operation since the applications are sitting on the top of the DBMS and operating system as indicated in the Figure 2.3.

As can be observed in Figure 2.3, upgrades to any one component of the enabling software base can cause a ripple effect on the applications. DBA is responsible for ensuring that software upgrades and vendor bug fixes are introduced in a controlled, non-disruptive manner. The DBA must make sure these upgrades are performed in the test and development machines first before they are installed in the production system. This will allow all applications to retest their programs.

The DBA is also responsible for ensuring all documentation related to upgrades, patches, schedules, etc. are available to users and developers.

Database maintenance also includes application, program, or module upgrades or enhancements. The DBA is responsible for installing these changes in the production databases after they are fully tested in the test database. There are also other maintenance tasks that the DBA has to

Figure 2.3 Layers of Software

do regularly, for instance, the regular shutdown and startup of the DBMS in order to perform database reorganization.

A check list of major DBA responsibilities in this category is presented below:

- ensures that software upgrades are introduced in a controlled, non-disruptive manner
- installs DBMS software and its upgrades and integrates it with the operating system [Tool]
- liaises with those responsible for testing to ensure that software upgrades are fully tested and suitable for inclusion in the production environment
- maintains a history of all changes to the software [Tool]
- ensures all documentation related to upgrades and new releases are available to users, developers
- implements database reorganization, changes, and application enhancements [Tool]
- performs maintenance of database objects such as building command files to recreate database objects such as user grants, indexes, etc. [Tool]

2.1.3 Security Administration

One of the main responsibilities of a DBA is the maintenance of database security. Security is the protection of the data in the database against unauthorized dictionary alteration or destruction [DATE 81].

More specifically, the DBA is responsible for who has what kind of access, to what information, in the databases. In a simple environment, this may be a trivial duty. However, in a situation where there are hundreds of users accessing the database, the issue of security management becomes much more complex. The DBA needs to have a mechanism to handle this in an efficient manner. Usually after a privilege has been given to a particular user, there is no way to find out why it was given. Is it a necessary privilege to the user, or just an historical remnant of a past requirement or a mistake? Otherwise users may end up with privileges which they do not require, presenting a possible security breach. Or they may not be given sufficient privileges to do what they are tasked for, causing operational difficulties [JAQU 87].

Hence the DBA should clearly define the security management strategy for his production database so as to protect an organization's data resource. The DBA is responsible for enrolling users and assigning the required database privileges and quotas. The DBA also needs to

revoke privileges and drop users and their related database objects when necessary. To protect the database, the DBA must review the privileges granted and decide on the audit actions on database and database object access as necessary.

A check list of the major DBA responsibilities in this category is presented below:

- manages user registration and its maintenance (drops user, revoke access, etc) [Tool]
- reviews, evaluates access controls established by the developer for a given application
- grants and revokes application access privileges to application users, administrators and developers [Tool]
- monitors the security and integrity of the corporate database [Tool]
- enables or disables audit of users, applications, user classes, privileges, etc. [Tool]
- automates easy removal of user objects if required [Tool]
- generates reports on security information related to users, applications, privileges, etc. [Tool]
- monitors the currency of enrolled users, privileges [Tool]

2.1.4 Operational Monitoring and Control

The DBA operational management activities are to ensure the smooth ongoing operation of the DBMS. This includes defining a strategy for backup and recovery, ensuring availability of on-line and batch processing opportunity, fixing DBMS related problems, analyzing DBMS status and trouble reports [ORAC 88]. Monitoring tools play a major part in helping the DBA with analyzing the DBMS status and locating trouble areas.

The ability to recover quickly from hardware or database failure is an important responsibility of the DBA. As a DBA, recovery strategies need to be anticipated so that various forms of failure can be handled with as little impact on database users as possible. Along with different types of failure (e.g. hardware failure, system software, application program) there are different recovery options [BEST 84]. Hence the DBA must be familiar with the various types of backup mechanisms and the corresponding recovery options. The DBA must determine a backup scheme that weighs the database administrative costs and the recoverability desired. Database recovery is dependent on the type of failure that has occurred and the files that may have been affected.

The DBA is the person who decides if recovery is necessary and how it should be done. If the recovery performed is unnecessary or

inappropriate, valuable resource will have been wasted and the database will have been denied to users when it did not have to be.

The DBA also performs ongoing monitoring of database usage for both on-line and batch processing, monitoring for exceptions to normal patterns of usage. Using proper monitoring tools, the DBA can view ongoing use of the database. For example, the DBA can find out the following information:

- the user names of current database users
- which programs they are running
- which users are waiting for other users
- which tables they are accessing
- a summary of I/O and I/O bottlenecks
- current and cumulative statistics about database usage
- number of reads and writes to each data files

These valuable information can help the DBA to identify tuning required to support increasing number of concurrent users; identify heavily used applications that may benefit by tuning; locate any locking problems; pin-point heavily used tables for I/O balancing over different physical devices to resolve any I/O bottlenecks and improve performance; tune DBMS parameters; provide inputs to the computer system manager for operating system tuning; etc.

One other primary day-to-day operational task of a DBA is maintaining knowledge on every aspect of the data in the database. From application information to storage allocation, from index design to data analysis, from users to responsible parties the DBA is expected to "know it all." This information is maintained and can be accessed in the DBMS's data dictionary. By analyzing these valuable data on a regular basis, the DBA can maintain a comprehensive data administration strategy, monitor the storage trends, and insure user compliance with database guidelines.

The DBA is also responsible for correcting any DBMS related problems which may entail incorporating temporary solutions so as to keep the DBMS available prior to performing corrective maintenance .

A check list of the major DBA responsibilities in this category is presented below:

- ensures the smooth ongoing operation of the DBMS with the help of monitoring and reports [Tool]
- defines a strategy for backup and recovery, and automates the required backup and recovery methods if possible [Tool]
- monitors and analyzes the database status as closely as possible so as to ensure the production applications as well as the DBMS are operating in an optimal manner [Tool]

- provides procedures for users to access different databases [Tool]
- ensures the availability of on-line and batch processing opportunity
- performs ongoing monitoring of DBMS usage for both on-line and batch processing and looks for exceptions to normal patterns of usage [Tool]

2.1.5 Space Usage and Monitoring

The most natural characteristic of a database is growth. Databases grow through the addition of more data of the kind it already contains and by the addition of new kinds of data through the implementation of new applications. Hence one of most important functions of the DBA is insuring that the potential growth of the database is anticipated and controlled [LYON 76]. The growth cannot be permitted to compromise the integrity of the database through unnecessary incorporation of redundant data. At the same time, administrative procedures that are created by the DBA must not discourage users from employing the database for new or extended applications. Hence the DBA must project the space requirement in the database system to meet the needs of all applications and users.

The need for growth must be encouraged and supported by the DBA in order to satisfy new requirements for information. Growth is accommodated from the time of initial design throughout the life of a database. The DBA must examine all proposed applications to insure system resources (storage medium, memory, etc.) can accommodate the addition of new applications. Improper consideration of space requirements can result in the most serious of all performance issues; it can result in the halt of some database operations within a database.

Running out of space in a database can result in significant amounts of downtime.

A second aspect of growth is the physical changes in the database files. The gradual addition of records through the normal conduct of the function of the organization eventually exhausts the allocated space or creates data fragmentation (records reside in non-continuous space).

Benchmarks showed that performance usually decreased as the fragmentation increased [LEFF 89]. So the DBA must regularly monitor and analyze the space utilization within the database to avoid running out of space during the peak hours of operation and to detect the fragmentation of the database files with the intention of improving database and application performance.

The other area of physical changes in the database files is the creation of non-reclaimable space in the database due to gradual deletion of records by the applications. The DBA has to detect this and performs the rearrangement of existing data to reclaim usable space if possible.

The DBA should also detect the over allocation of space by a application to reduce the unnecessary usage of excessive storage space.

The DBA is also responsible for the implementation of the database architecture according to the storage method of the DBMS. This may

involve the spreading of database files across different storage media to achieve better performance.

Obviously, the effective management of space usage within a database is an important part of the job responsibilities of any DBA. In some installations, the management of database space can consume as much as twenty per-cent of a DBA's time [LEFF 89]. Thus DBA tools are important vehicles for the DBA to streamline, automate, and improve the above functions related to space management. For example, tools can help the DBA to automate these functions: estimation and calculation of entire database storage requirements; estimation and calculation of individual database object storage requirements; monitoring and online analysis of overall database space usage and space availability; monitoring and online analysis of storage usage on an individual user and database object basis; dynamic resizing and reallocation of individual database objects; dynamic resizing of databases; etc.

In all, the DBA is required to monitor and control the database in terms of storage capacity planning and projection and to provide reports on trend-based analysis of space usage so as to prevent potential problems before they occur, to take corrective actions to free up space, and to carry out performance enhancements. The space management function is an important part of the job responsibilities of a DBA.

Failing to pay enough attention to this function can result in severe performance and availability problems [LEFF 89].

A check list of the major DBA responsibilities in this category is presented below:

- forecast the storage requirements for the DBMS and all applications [Tool]
- provides tools for the developers to calculate data and index storage requirement for their applications [Tool]
- designs physical database structures that meet the needs of both end users and the computing facility
- implements database structures within the storage method of the DBMS
- monitors and analyzes storage utilization to determine and assess potential problems such as data fragmentation, chaining, space limitation, etc. [Tool]
- generates reports on disk space and user utilization, free space summary, space utilization by database objects, and trend-base analysis reports on these issues [Tool]

2.1.6 System Resource Usage Monitoring, Performance Tuning

One important function within the database administration function is the responsibility for performance measurement and tuning of the database management system and the applications.

The performance of the database should be monitored and statistics collected and analyzed. Performance monitoring is an ongoing process aimed at uncovering operational problems with production systems or providing early warnings of potential performance problems, particularly when the volumes of data grow dramatically or when major modifications are made to existing systems. Monitoring database utilization can help the DBA to identify systems that consume sufficient resources to possess potential for optimization.

Monitoring also includes the gathering of operating system resource utilization details such as disk usage, CPU usage, and I/O usage. Performance statistics can be used to predict future growth and inform management when the current hardware is likely to be saturated. Also growth of the disk space needs to be monitored so that long-range planning of hardware acquisitions or replacement is done in an orderly fashion. These operating system related activities require the co-operation between the DBA and the Computer System Manager or System Manager.

Again, monitoring tools play an important role in helping the DBAs to capture some of the statistics for database, operating system, performance, and resources such as CPU usage, I/O, memory, file accesses, etc. These tools also include operating system monitoring tools which may be provided by the System Manager.

The DBA should also investigate complaints from users about response time and determine the problem areas to be addressed. The DBA should also work very closely with the operating system support staff (System Manager) to overcome any problems.

The DBA must monitor the performance of the application systems and cannot always rely on users to describe their applications accurately or to interpret the symptoms of poor performance correctly. An improperly designed database application program can quickly consume all available I/O, and can downgrade the performance of the entire database system.

When tuning an application system, the following tuning points should be considered [RUSS 84] :

- DBMS and Operating System Installation
- Database Design
- Program Design
- Application Testing

The DBMS itself has certain parameters that can be altered, and the proper adjustment of these parameters may be critical to the performance of the DBMS. The DBMS tuning is performed by the DBA. The operating system also has some parameters that are crucial to the efficient performance of the DBMS. The System Manager is responsible for tuning some of the operating system parameters based on the information provided by the DBA. From the the above, it is noticeable that there is a close relationship between the DBA and the System Manager. A brief comparison on the duties and co-operation is worth discussing here.

As discussed earlier, the DBA is responsible for any DBMS-related activities such as installing and upgrading of DBMS software, databases, database applications; setting up of database accounts for database users; monitoring database activities and performance; perform DBMS tuning; carry out database backups and recovery; etc. Whereas the System Manager is responsible for any operating system related activities such as installing and upgrading of operating system software; setting up of computer user accounts; monitoring operating system activities and performance; perform operating system tuning; carry out operating system backups and recovery; etc.

Even though there is separation of duties between the DBA and the System Manager, there also exists co-operation between them especially in the area of tuning the operating system for the optimal performance of the DBMS. This tuning by the System Manager required the inputs from the DBA. These inputs may be: requirements of the DBMS in that operating system; DBMS vendor recommendations based on the database performance statistics; etc. The System Manager needs input from the DBA to set up the proper computer user account requirements for using the DBMS software. Sometimes the DBA may need to tune the DBMS based on the information supplied by the System Manager. The information may include disk I/O bottlenecks, memory usage statistics; etc. Thus the DBA can avoid the I/O bottlenecks by moving the DBMS physical files around to better balance the I/O.

Now let us go back to look at other aspects of application tuning. Database design is important to ensure appropriate logical designs are produced which are easily and efficiently implemented in the DBMS. Program design should be reviewed to ensure that efficient access is made to the database using as few I/O accesses as possible.

Application testing can detect errors in the database design or program design when response time or elapsed time of a job does not match expectations. At this stage it may be too late to redesign the

logical database, but program design can be examined or modified. Some methodologies propose that volume testing and response testing of the database design be done early, before major commencement of coding, in order to get an acceptably performing design. The DBA should also liaise with the developer during the testing phase to document and analyze expected resource usage for a given application.

A check list of the major DBA responsibilities in this category is presented below:

- collects and analyzes DBMS and operating system resource utilization details and provides early warning of potential performance problems [Tool]
- monitors and gathers database utilization statistics to identify applications that consume sufficient resources to possess potential for optimization [Tool]
- performs tuning on applications, DBMS, and operating system as necessary
- investigates complaints on performance issues and determines problem areas to be addressed
- generates reports on systems's activities, condition of the database, usage of resource and efficiency of their utilization [Tool]

- looks for early warnings regarding poor space management, resource contention, poor memory utilization, uneven load across storage devices, I/O problems, locking problems [Tool]

2.1.7 Application Development Review and Evaluation

It is important for the DBA to establish a standard DBMS environment for his organization. The DBA has to set up different databases in response to user needs. Of course the DBA also needs to consider the availability of resources. Typically one can find three types of database environment: production, development, and test in an organization. Rules are different for the three types of database environment and there must be a clear separation among them and ideally they should be on separate machines.

The development database is mainly for the developer to code and debug the development application so as to achieve the expected functionalities. When the development application is completed, it is moved to the test database for intensive testing.

The test database is mainly for development applications to perform activities such as volume testing, acceptance testing, CPU testing, etc, within the framework and in compliance with the organization's methodology. Ideally the test database set up (database and operating system parameters) should resemble those of the production database. The test database is also used by the applications to test their upgrades, changes, enhancements, etc. Once the tested application meets the user

and DBA requirements or expectation, it is then ready to migrate to the production database.

In a production database environment, as the name implies, only the production systems are allowed to be there. The database administrator is the custodian of the corporation's data [MART 77]. Being the custodian of the production database, the DBA allows a new application to go into production environment only after the application meets certain criteria such as proper database design, efficient coding, etc. This will ensure the new application will not affect other existing production applications.

The DBA is also responsible for installing new application and upgrades to existing applications in the production database according to agreed-to-procedures. Basically the DBA should have complete control over the database objects in the production database.

The DBA should also prevent untested programs from accessing the production database. For protection of the production database, a program is only allowed to run against a test or development data base while it is in programming and testing. This can be done through the software configuration control mechanism.

The DBA should ensure all necessary tools and utilities are installed and available within a given database environment.

In a typical DBMS environment, all applications share a pool of system and database resources. A poorly designed application can easily affect all other applications in the production environment. Hence it is important for the DBA to be involved in the different phases of the application development process. A typical development project consists of the following phases and their products [RUSS 84] :

1. Feasibility Study	- Feasibility Report
2. System Analysis	- System Proposal
3. System Design	- System Design
4. System Specification	- System Specification
5. Programming	- Tested and Documented Programs
6. System Testing	- User's Acceptance
7. Implementation	- Final Acceptance and Review

Each of these phases should be reviewed by the DBA and then become part of the permanent documentation of the system, allowing other database administration members to review design criteria and their relevance to the current system. The feasibility report allows the DBA to gain an appreciation of the scope of the application and the likely resource requirements such as hardware, resource, time, and money. The system proposal should be examined to see if the system is functionally possible. It also allows the DBA to understand the overall approach for

testing, acceptance and implementation and ensures they follow the DBMS standard or company guidelines. Any data analysis, any data model, and any other major database-related design should be reviewed by the DBA. This is a critical control point during the design process. Large organizations require a signed acceptance from the DBA before proceeding to program coding. Program design should be checked and the database access paths checked for efficiency. The programs or modules which will be executed most often should be examined closely, since the response time is dependent on the way the program accesses the database.

The DBA should also liaise with the developer during system testing and performance testing and provide developers with tools with which to assess performance and with advice as to how performance may be optimized.

By being involved in the above development phases, the DBA can ensure standards, such as naming conventions and DBMS guidelines, have been applied to design and program functions. The DBA can also screen out performance problem before the application goes into production. Moreover, the DBA can estimate the impact of a new application on the existing production applications.

During implementation, the DBA should install the application using the application objects provided by the developer. The DBA also is required to closely monitor the application during the implementation phase.

Since the DBA is the custodian of all production applications, any new versions (major enhancements) or release (maintenance, minor enhancement) of an application, has to be tested in the test database before it is allowed into the production database. The DBA should also have a record of all changes to all applications in the production system.

The DBA should also provide guidelines and utilities for migrating applications from development to testing, and from testing to production.

A check list of the major DBA responsibilities in this category is presented below:

- establishes shared data in different databases [Tool]
- ensures all necessary software tools, utilities are installed and available within a given environment
- reviews reports from each phase of the development project and makes recommendations as necessary
- reviews codes for new applications and its potential impact on the existing applications before approving its inclusion in the production database

- keeps track of all applications in the production database by capturing information such as application acronym, owners, responsible parties, system resources utilization, software tools, billing information, etc. [Tool]
- enforces application program compliance with DBMS standards and conventions (e.g. database objects naming conventions) by providing online information if possible
- records upgrades to the production systems by employing software change control practices [Tool]
- assesses impact of changes to data types or other system components by using utilities or CASE tools [Tool]
- provides utilities for migrating applications from one state to another (e.g. test to production) [Tool]
- installs application and its upgrades in the production database according to agreed-to-procedures [Tool]

2.1.8 Dictionary Administration

Every DBMS vendor provides a data dictionary or system catalog which is also a component of an integrated database management system. The dictionary contains and maintains the definition and description of the data and the database. The data dictionary describes the information about the data (called metadata), and the DBA has to ensure the accuracy and integrity of DBMS data dictionary.

However the vendor's data dictionary may not meet all the requirements of an organization. The necessity for an expanded data dictionary becomes increasingly important as an organization integrates its systems in a database. As the user community grows as well as new applications go into production, it is increasingly critical for the DBA to be able to track systems information, user information, performance information, etc., through a data dictionary. The data dictionary provides a vehicle for defining standards, identifying sources of discrepancies, and determining responsibility for data quality. The data dictionary can also be used to aid in the control, management, and documentation of applications, DBMS, development phases, security management, etc.

As the responsibilities of the DBA contain various important tasks, it is crucial for the DBA to use a data dictionary to keep track of

additional valuable information that is not found in the DBMS's data dictionary. This will make the life of the DBA easier. By storing, and making available all information about a database in a standardized data dictionary, the DBA is able to perform monitoring, control and administrative activities in a more effective and efficient manner.

A check list of the major DBA responsibilities in this category is presented below:

- implements a dictionary that correlates and defines the information that is important to an enterprise and to the DBA functions
- maintains the data dictionary and its schema [Tool]
- evaluates and analyzes information captured in the data dictionary [Tool]

3.1.9 Other DBA Administrative Activities

For large organizations, it is very important for the DBA to communicate with the users and programmers on a regular basis. Any major planned or implemented changes to the system or any of its support software, should be communicated through a regular newsletter or other means.

Other important DBA functions include the continuing education of the users, programmers, and project leaders. This can be done through technical bulletins, user group meeting, seminars, formal education and reviews.

The DBA should also liaise with operations staff to ensure synergy between DBMS, operating system, network and hardware.

The DBA should also provide feedback to management regarding availability, security and performance of the DBMS. The DBA also has to continually monitor, evaluate and improve the quality of the DBA toolset.

Internally, the DBA acts as an inhouse consultant to users and programmers on DBMS usage. Externally, the DBA is responsible for the liasing with DBMS vendors, and other DBA's of different organizations.

2.2 Survey of Database Administration Tools

Managing a complex DBMS environment can be a difficult and time consuming task for the DBA. The DBA must maintain the availability of the DBMS, be able to recover data from hardware or software failure, manage security, source code, users, applications and storage capacity for the DBMS environment, perform regular backups and performance tuning, and install software, etc. Hence the task of managing a DBMS can quickly become unwieldy unless the DBA can automate some of the day-to-day functions to assist in accomplishing the complex DBA mission in as efficient a manner as possible.

The numerous tasks performed by a DBA are often painstakingly complex, tedious and challenging. While these efforts are necessary, they need not be a burden. Using automated or semi-automated tools, many of the DBA duties can be made easier and much less time-consuming. This in turn, provides the DBA with more time for other important job functions such as analysis of operational and performance information. In analyzing the basic responsibilities of a DBA in Chapter 2, there are many functions which readily lend themselves to automation due to their repetitive, time-consuming nature.

With the provision of automated tools, performing certain DBA functions can be made more efficient. Through automation of some of

the DBA responsibilities, the monotony of these functions can be eliminated and the results of these tasks can be much more thorough and reliable.

A study of DBA tools for a representative set of Relational DBMS (RDBMS) was carried out to examine their features. The examined tools were related to the ORACLE, INGRES, SYBASE and DB2 relational database management systems. A brief set of findings of the DBA tools for these DBMSs are given in the following paragraphs. An overview of each of the surveyed tools is given in Appendix A.

Findings for Surveyed DBA Tools

Each of the surveyed ORACLE DBA tools only service a single database (e.g. DBA Companion, Database Analyzer) or connected distributed databases (e.g. SQL*DBA, easyDBA) at a time. They only automate a small portion of the overall DBA functions. For example, DBA*Master automates the DBA functions in the category of System Resource Usage Monitoring and Tuning. One other point is that one could not extend these tools (e.g. SQL*DBA, easyDBA) to meet changing environments and new requirements.

The surveyed INGRES DBMS Server only automate the areas of Security Administration; System Resource Maintenance and Tuning of the DBA functions. Again it only services a single database at a time.

The surveyed SYBASE DBA tools also performed a subset of the overall DBA functions and they also only service one database at a time despite the distributed database capability of the tool (e.g. SA Companion).

The surveyed DB2 DBA tools again performed a subset of the overall DBA functions such as: data set analyzing, application migrating, catalog management, security management, etc. Again these tools only service a single database.

Overall, all the surveyed DBA tools only work with one type of RDBMS. For instance ORACLE tools only work with ORACLE databases, SYBASE tools only work with SYBASE database, and ORACLE tools will not work on SYBASE databases. They only allow the DBA to work with one database at a time even with their distributed database capabilities. The surveyed DBA tools mainly work with the RDBMS vendor's system catalog and the tool's own data dictionary. Thus one could not add functionalities to these tools or alter the tool's data dictionary to capture other information that the DBA would like to see or monitor. Finally the DBA may need multiple copies of the same tools installed in different nodes if the database are not connected.

2.3 Summary

The DBA functions consist of an array of administrative, monitoring and tuning activities. Some of these activities can be assisted by reviewing the information residing in the vendor's data dictionary or system catalog which is one of the most important objects in a Relational Database Management System (RDBMS). The RDBMS system catalog is a set of relational tables or views to be used as a read-only reference guide about the database. It contains a valuable source of information for the DBAs as well as other types of database users. The system catalog includes application information, security control, auditing details, space usage information, user profiles, information for performance and tuning analysis, backup and recovery information, etc. A typical RDBMS system catalog may contain hundreds of relational tables and views.

In order to successfully manage a RDBMS, the DBA must have an in-depth knowledge about the RDBMS system catalog, has to know what types of information are contained in the RDBMS system catalog and must be able to join the information together from the RDBMS system catalog so as to make quick and important decisions. To achieve these goals, the DBA has to rely on automated tools which provide interfaces to access information contained in the system catalog.

The DBMS vendor-supplied or commercially available DBA tools allow the DBAs to facilitate the execution of some of the DBA activities, but directed at a particular database. In an environment where the DBA has to manage multiple databases, the DBA is forced to use the tools to access different databases one by one. The DBA ends up duplicating many DBA functions at each database. Frequently, multiple license costs for the same tool are also incurred. Hence a better mechanism should be examined to help the DBA to deal with the multiple database environment.

CHAPTER 3

DATABASE ADMINISTRATION

IN A

MULTIPLE DATABASE ENVIRONMENT

During the 80's the number of applications which use database as the repository for corporate data has increased dramatically.

Simultaneously we have witnessed the proliferation of communication networks designed to permit the exchange of data amongst computer nodes in those networks.

These two factors have given rise to the third phenomenon: multiplicities in types of databases and types of computing hardware upon which they operate. We see environments with centralized databases operating on many different types of computer systems.

This results in a new set of challenges for those who are charged with administering databases in this environment. This section explores the types of environments in which today's DBA must function and defines the problems which the DBA faces. Finally, this section describes a proposed solution which can facilitate the effective administration of databases in this new environment.

3.1 Overview

Centralized Database Environment

A centralized database is a collection of data which resides in one computer. Other sites of the enterprise may access the database from remote locations by logging on to the central computer through the computer network. This type of environment is found in many small organizations who wish to pursue a strategy of centralized processing of these data. A diagram of such an environment is shown in Figure 3.1.

Another variant on this environment is the client-server architecture, where PCs are connected to the centralized database. A client is an application requesting data from another node within a computer network. It may be a node with a user interface or an application front end tool. Clients can be PCs, minicomputers, or mainframes. A server is any node with a database, from which data may be requested. Servers must support multi-user operating systems, running multi-user DBMS [ORAC 88]. In this environment, the front end programs are run at the PCs while the data are stored centrally in the server. This is found in many microcomputer-based Local Area Network configurations and is gaining acceptance and use. A diagram of such an environment is shown in Figure 3.2.

Figure 3.1 Centralized Database Environment

Figure 3.2 Client-Server Centralized Database Environment

The centralized database environment is characterized by the existence of a centralized database administrator who has the responsibility for administrating the database. In this type of environment, the DBA needs only to deal with one database. The system catalog, or database dictionary, contains the information regarding the status of that database. This is the repository for information about the database. The DBA examines the system catalog of the centralized database in order to manage and monitor the status of the centralized database. DBA tools for handling a single database are common and are commercially available from both the database vendors and from the third party software enterprises.

Distributed Database Environment

"A distributed database is a collection of data which are distributed over different computers of a computer network. Each site of the network has autonomous processing capability and can perform local applications. Each site also participates in the execution of at least one global application, which requires accessing data at several sites using a communication subsystem" [CERI 84]. A diagram of this environment is shown in Figure 3.3.

In other words, a distributed database is a set of databases stored on more than one computer in such a way that users perceive the data as a single "large" database (global database) when in fact it is several "smaller" databases (local databases). These databases are linked together by vendor-provided software (database links).

It is important to note that each computer runs the software to access the other databases; there is no central software which coordinates all participating databases. This provides the advantages of site autonomy and avoids a potential central point of failure.

Figure 3.3 Distributed Database Environment

Note: Components 'A1',.....'A7' of the database 'A' are distributed across the Network.

With distributed databases it is possible to identify a hierarchical control structure based on a global database administrator, who has the overall responsibility of the entire database, and on local database administrators who have the responsibility (maintain, start, backup, and so on) for their respective local databases. However, it must be emphasized that local database administrators may have a high degree of autonomy, even to the point where the global database administrator's position is not identified and the intersite coordination is performed by the local database administrators themselves. This characteristic is usually called site autonomy. Distributed databases may differ very much in the degree of site autonomy: from complete site autonomy without any centralized database administrator to almost completely centralized control [CERI 84].

When multiple computers are connected by a network with the intention of allowing users and applications access to multiple nodes, several new DBA concerns arise. These concerns can be classified in the following categories:

- installation decisions (where to install which software and where to locate which data)
- guaranteeing data security and integrity across the network

- guaranteeing good system performance on all nodes and across the network.

The global DBA has to resolve these concerns with the local DBAs.

In this type of database environment, the global DBAs have to deal with multiple databases and system catalogs. The global DBA has to log on one computer and sign on to each individual database through database link so as to monitor the status of each database in the database network. The local DBA has to log on a local computer and sign on each individual local database so as to monitor the status of the local databases. In order to perform a common function on each database (e.g. space availability at each database), the DBA has to examine each system catalog in turn. In doing so, the DBA repeats the same action at each database. Some of the surveyed DBA tools provide capabilities for handling distributed databases. Again these tools only examine one system catalog at a time.

Multiple Database Environment

A multiple database environment consists of a set of databases which are resident at the same or different sites of a computer network. In other words, the multiple database environment is a distributed implementation of centralized databases. There is no database connection amongst the local databases and thus they are sometimes referred to as multiple databases or multiple separated databases. Each site by itself may have multiple databases which are resident at more than one computers connected to a local network. A schematic of such system is given in Figure 3.4.

Multiple databases are required as a result of user needs such as the requirement of development, test, and production databases within an organization (discussed in Section 2.1.1 and 2.1.7) or as a result of heterogeneous database and/or hardware environment which is discussed in the next section. It is also anticipated that increasingly, a local database will be required on workstations to manage the information located on the workstation via introduction of active repository-based approaches to information management. One of the common actions on the multiple databases is to move applications from one database to the other: development to test, test to production. DBA tools are required to facilitate this type of migration. Some of the surveyed DBA tools

Figure 3.4 Multiple Database Environment

Note: Various databases are located in each node of the Computer Network.

do handle this type of migration.

Multiple databases are characterized by complete site autonomy. Each site has its own local database administrator who is responsible for the administration of the local databases. In this type of database environment the DBAs have to sign on each database separately (due to the non-existence of database link) so as to review the information and status of each database. In doing so, the DBAs repeat many similar actions at each database in order to accomplish the same task.

All of the surveyed DBA tools only provide capability to handle a single system catalog at a time. For DBAs who need to manage multiple databases, even at a local site level, these tools need improvements. One such improvement is to be able to examine multiple system catalogs by signing on a single database.

The main difference between the multiple database environment and the distributed database environment is that the databases in the multiple database environment do not communicate with each other whereas databases in a distributed database environment are parts of a single logical database, hence they may communicate with other.

In practice, a typical large organization may have a combination of multiple database and distributed database environments. For instance, the production databases within an organization are all linked while the

development and test databases are not linked to the production databases. As another example, it is anticipated that local workstation databases acquired for managing the local information space will also often be used in addition to participate in distributed database applications as well.

3.2 Issues in Database Administration

In the 90's, many medium to large size organizations are faced with challenges associated with the introduction of new technology such as communications networks, new generations of databases, multiple hardware platforms and pervasive use of powerful workstations. The challenges may be further examined in terms of a number of discrete problems which face the DBA in particular. We identify these problems by investigating the following issues:

- Focus of Current DBA Tools
- Proliferation of Databases
- Multiple Sites
- Heterogeneous Environment
- Data Complexity
- Growth and Changing Operation Environment

Each of these issues are briefly discussed in this section. We then examine a real example of this complex environment.

3.2.1 Focus of Current DBA Tools

As described in Chapter 2 of this document, the DBA functions consist of an array of administrative, monitoring and tuning activities directed at a particular database. The DBMS vendors and third party software vendors have provided a basic set of tools with which to facilitate the execution of some of these activities, but these are directed for the most part at a single instance of a particular database. Therein lies the one of key problems. The reality of the rapidly emerging networked environment is that the DBA must routinely deal with multiple databases, but must attempt to do so using tools and techniques which do not recognize this fact. Figure 3.5 illustrates the focus of current DBA tools.

The DBA tools commercially available today do not provide for the effective administration and monitoring of multiple databases. This is confirmed from the survey of DBA tools described in Section 2.2. Each of the surveyed DBA tools only service a single database or connected distributed database. However they do not provide the capabilities required to manage multiple databases. This forces the DBAs to duplicate many DBA functions at different databases by frequently signing on and off. Moreover, these tools are also quite limited in the functionality they provide. They usually automate a small portion of the

overall DBA functions we discussed in Chapter 2. This calls for a DBA tool to manage this type of common database environment. With proper support tools, the DBAs have the ability to manage the steadily increasing number of activities required to administer and support today's complex applications.

Figure 3.5 Focus of Current DBA Tools

3.2.2 Proliferation of Databases

In the 90's, enterprise-wide computer networks will become a normal and common way of interconnecting computing devices. The number of databases on this computer network can only increase. This will occur because databases will tend to be established for local workstation information management and support purposes. In addition, we will see increasing introduction of applications which will use the power of the network to access other company-shared databases as well as the application-specific databases.

The nature of these databases includes production, test and development databases which are discussed in Section 2.1.1 and 2.1.7. Some of the production databases are connected by a database network (distributed database) and some of them are not (multiple databases). This is also true for development and test databases. Moreover, there is a clear separation amongst the production, development and test databases. Hence there exists special purpose distributed databases as well as multiple databases.

Moreover, due to the economies of hardware, departmental production, development or testing machines are becoming increasingly popular, as are workstations with local database applications. This has also contributed to the proliferation of databases.

The proliferation of databases at a single site is illustrated in Figure 3.6.

For a given database, it may have more than one occurrence. For example, the Human Resources application accesses a Production Database. There may also be a test copy of that database used for testing enhancements to the Human Resources application. There may also be a separate development copy of the database used by application developers as the environment in which to develop a new version of the Human Resources application. At any given point in time all three databases could be in use. Each copy of this database - Production, Test and Development would have different requirements in terms of DBA support and administration.

Figure 3.6 Proliferation of Databases

3.2.3 Multiple Sites

As networks increase in scope, and span geographic locations (for example, branches of a company spread across the country all connected through one enterprise-wide network), a company will likely wish to establish the DBA function in multiple locations. This will give rise to an associated problem, that of coordination and standardization amongst the DBAs. Some examples of these are:

- to co-ordinate the version control of RDBMS products among distributed databases;
- to standardize RDBMS environments amongst all distributed or multiple databases to allow easy portability of applications across the company network;
- to provide tools to permit the efficient migration of applications from one database to the other. For example, from development to test, from test to production databases, or from a test workstation to many end user workstations connected to the network.

Again, the multiple sites environment contains multiple database as illustrated in Figure 3.7.

Figure 3.7 Multiple Sites

3.2.4 Heterogeneous Environment

In the 90's we will see an increase in the types of databases which are installed on the company-network. There will be a move toward a heterogeneous environment in terms of hardware and software. One of the current trends from the hardware vendor is the availability of free run-time RDBMS license for each purchased processor. It is hard for an organization to keep users from using these free DBMSs. This trend contributes to the heterogeneous environment in the 90s. This type of environment is illustrated in Figure 3.8. The vendors of these various databases have constructed the DBA tools such that they only deal with that particular vendor's database. Therefore the situation begins to unfold where the DBA uses Tool A to look at vendor A's database and Tool B to look at vendor B's database and so on.

Figure 3.8 Heterogeneous Environment

3.2.5 Data Complexity

The day-to-day activities being performed by the DBA will become more complex as the successive implementations of the vendor's database software becomes more complex. Even today we see the major RDBMS vendors indicating that they will be adding object-oriented extensions to the relational model, they will also be extending the model to encompass rules-based processing, expert systems concepts, artificial intelligence, etc. [STON 90]. Moreover in recent years, relational database systems have begun to handle unstructured and non-standard data types such as text and graphics. Difficulties arise in the efficient handling of these data types (storage, searching, indexing, manipulating, etc.) [MCGO 90]. For example, with the introduction of large complex data types to the relational model, we will see databases which are physically (and purposely) fragmented to accommodate the storage of various data types on the most appropriate medium. In other words the columns of a row within a particular database could be stored on different media. These complexities simply cannot be adequately administered and monitored by today's DBA given the narrow focus of the current DBA tools. The data complexity is illustrated in Figure 3.9.

Figure 3.9 Data Complexity

3.2.6 Growth and Changing Operating Environment

One area that the DBA may have no control over is the growth and change of the computing environment. Some examples of these are:

- new nodes containing new databases are added to the network;
- new computer types introduced to run existing databases;
- another node (or new computer type) introduced to run a new database type;
- "twinning" of databases on a node (or nodes) during the period of a version upgrade;

The DBA must be able to handle these situations in such an effective manner that it is routine (because on a large network it will be!).

Finally, as more and more mission-critical applications and data take advantage of database technology, the administration, monitoring and tuning aspects of the database take on ever increasing importance to the correct (and competitive) operation of the enterprise.

3.4.2 DBA Tools for Multiple Databases

A consolidated dictionary is useful only if the DBA can get at it and use it. This is true of any collection of any kind of data. So DBA tools are required to access, analyze, and report on the collected data. These tools access the consolidated dictionary and thus permit cross-analysis among different databases.

This study examines some of the repetitive functions that the DBAs have to perform at different databases. DBA tools are then evaluated to automate these functions so that the tools can operate on all the data collected from different databases as well as sending command operations back to different database. Most of these functions are analysis, monitoring, and reporting functions. Some of these functions which are identified in Section 2.1, may include:

- analysis of space availability and space usage
- monitoring the currency of enrolled users
- reporting on security information
- reporting on data fragmentation and chaining
- monitoring and reporting on database files that are almost full
- reporting on applications information
- reporting on database information
- performance of functions to enroll, delete and alter database users.

3.3 Problem Definition

In this Section, we discuss the problems relating to the management of multiple databases, by looking at the following issues:

- Problems with Managing Multiple Databases
- Problems Facing the DBAs

Then we examine an real-world example of multiple database environment.

3.3.1 Problems with Managing Multiple Databases

Each system catalog contains information only about the database to which it is attached and this makes it difficult for DBAs who have to manage and support more than one database.

Many operating environment need to support multiple databases such as production, test, development and multiple workstations with local databases. Database administration quickly becomes over-whelmed when more and more databases are required due to user needs or other reasons such as different versions of RDBMS. The DBA has to deal with many issues in the production environments. If a production system is unavailable, or if database recovery is delayed, the user sustain a quantifiable loss. The DBA can easily spend much time in monitoring the production environments without spending time in the analysis of the status of other databases around him. Serious problems may arise if the DBA fails to monitor different databases such as the development databases or local workstation databases. The system development process may be hampered by failing to take proper actions much earlier.

In multiple database environments it is necessary to look at the RDBMS system catalog information contained in more than one system catalog. The simple approach would be to examine each system catalog independently. This is the approach used by most vendors. This forces

the DBAs to sign on and off from different databases in order to examine and analyse the information in different system catalogs. The DBAs may end up doing many duplicate functions at different databases and this would easily take away the DBA's valuable time in performing other major tasks. A typical schedule of the DBA in managing multiple databases today is depicted in Figure 3.10. With anticipated explosive introduction of local databases on workstations, this schedule will rapidly deteriorate to a completely unmanageable situation.

Figure 3.10 Typical Activities for DBA for Managing Multiple Databases

Note: There are not any database links amongst the databases.
The computers are linked by a computer network.

3.3.2 Problems Facing the DBAs

Due to the new environments in the 90's as we have discussed in the previous sections, the DBAs now face new challenges to manage, support and administrate the changing database environment. Some of the problems that are facing the DBAs are:

- Number of Databases
- Heterogeneous Environment
- Criticality of Performance, Availability
- Increasing Complexity

Each of these issues is briefly discussed in the following paragraphs.

Number of Databases

The DBAs at many organizations now face a number of problems. One of these is the increasing number of databases due to user needs and requirements. The DBAs have to manage the increasing size of the production databases as well as increasing activities on the test and development databases. This makes it necessary for the DBAs to replicate many routine DBA functions or activities at each database. To complicate matters, the DBAs also have to deal with new version of the RDBMS and this implies that new databases need to coexist with the old version. Hence multiple DBAs are needed to coordinate different activities and to standardize operations on these databases. The anticipated introduction of local workstation databases will make the situation untenable for the DBA with the current tools.

Heterogeneous Environment

The second problem that the DBAs have to address is the migration towards a heterogeneous environment. There exists different hardware and software (operating systems and network software) environments. This implies that multiple databases from various DBMSs are running on various hardware platforms. The proliferation of different DBMSs on various hardware platforms is going to add significantly to the work loads of the DBAs.

Criticality of Performance, Availability

The other problem facing the DBAs is that their day-to-day duties become increasingly important as more and more mission-critical applications using the DBMS are installed. The DBAs, of necessity will tend to focus on the production systems with little time left in serving the needs of the development, testing and workstation communities.

Increasing Complexity

The last problem that the DBAs have to face in the 90s is the increasing complexity of the RDBMSs which are evolving into object orientation and rules/knowledge based concepts. The DBAs will have to deal with activities associated with object management and knowledge management as well as database management.

3.3.3 Example

AECL Research Company (Atomic Energy of Canada Limited Research Company) is a crown corporation of the Government of Canada with its head office in Ottawa and two research laboratories: Chalk River Laboratories (CRL) at Chalk River, Ontario and Whiteshell Laboratories (WL) at Pinawa, Manitoba. All three sites have ORACLE RDBMSs running on a number of VAX nodes which are connected through the company-wide communication network known as RCNET. At each site, there are PRODUCTION, TEST, and DEVELOPMENT ORACLE databases running on different VAX computers. These computers are in turn called production, test and development machines. Moreover AECL Research also has other DBMSs running on various hardware. These products include INGRES, DATA MANAGER, BASIS PLUS, dBASE IV, etc. Local workstation databases have not yet been widely introduced, but their introduction is expected to occur in the near future. An overview of the organization of the database environment is illustrated in Figure 3.11.

Currently the database environment at AECL Research is a multiple database setup. There is as yet no database linking among the individual databases at the local or global level. There are many

Figure 3.11 Existing Database Environment at AECL Research

databases at each site serving application for specific purposes. Each site has a DBA to serve local interests. Each local DBA is equally burdened with the task of managing multiple databases such as development and test databases in addition to a few production databases. Anticipated introduction of local workstation databases would completely overload existing DBA capacity.

Let us examine one of the local site of AECL Research Company's database environment in detail. At Chalk River Laboratories there exists the following environment:

- ORACLE RDBMS running on VAX mainframe, micro VAXes, VAX stations, MACs and IBM PCs under the operating systems such as VMS, DOS, OS/2, etc.
- BASIS PLUS DBMS running on VAX mainframe.
- DATA MANAGER RDBMS running on CDC hardware.
- dBASE IV, etc. running on PCs.

Figure 3.12 illustrates the database environment at CRL.

From the above, it is obvious to notice that the database environment at AECL Research's CRL site is a typical multiple databases environment.

Figure 3.12 Multiple Database Environment at Chalk River Laboratories

Note: There are no database links amongst the databases.
The computers are linked by a local area network.
There are many PCs, MACs running ORACLE, dBASE IV.

The various hardware components are connected by a local area network at each site, interconnected into a wide area network, but there is no database linkage amongst the databases. However this is in the process of changing as AECL Research is now studying the linkage of few ORACLE databases at local sites and then plans to move further towards a company-wide distributed ORACLE production databases. Moreover, AECL Research is also in the process of introducing DEC's Rdb RDBMS into the VAX/VMS environment on a company-wide database. Anticipated introduction of an increasing number of Unix workstations with free INGRES databases may add further to the choice of databases. Thus the heterogeneous database environment is rapidly becoming a reality at AECL Research.

Moreover, due to the lower cost of hardware as well as increasing user requirements for management of structured information, more and more department computers are purchased with a DBMS running in it. AECL will also increasingly consider utilizing a local database to manage the local workstation information space in an active repository-based manner. Thus the number of databases at AECL Research is growing rapidly and significantly in the 90s. Furthermore, large data types such as graphics, images, expert system rules, etc. are required by the

scientists and other researchers in their databases. This adds to the data management complexity issue discussed in Section 3.2.5.

Hence in the 90s, AECL Research is moving strongly towards a database environment with multiple databases, distributed databases, heterogeneous databases and complex databases dispersed across a heterogeneous hardware platform. Tools are required by AECL Research's DSAs to managing such a complex and changing environment.

3.4 A Proposed Approach to Manage Multiple Databases

The DBAs require a toolset to help them deal with the complexities of administering an increasing number of databases. The toolset should accomplish some or all of the following:

- provide a single tool (or toolset) which can monitor and analyze the status of all databases on the enterprise network regardless of database type or the type of hardware upon which they operate;
- automate the routine, repetitive tasks that the DBAs perform;
- provide for frequent monitoring and notification of critical resource usage and performance management factors which can be dynamically defined to the toolset;
- provide for identification and registration of new databases within the enterprise regardless of the type (vendor) or location on the network or version, etc.;
- provide for extensibility of the data dictionary to allow for changing DBA requirements;
- provide for the extensibility of the toolset to allow for added functionality;
- recognize differing degrees of monitoring, control over different databases.

In order to accomplish or permit the above, the DBA toolset should consist of a centralized repository and a set of associated tools.

3.4.1 A Data Dictionary for Multiple Databases

In order to allow extensibility and be able to describe and control the consolidated data, a metadata dictionary should be explored to handle these. Ideally this metadata dictionary should conform to a standard and allow for portability. One of these standards is the Information Resource Dictionary System (IRDS) which is discussed in Appendix B and C.

In Appendix B, we discuss the importance of information resource management using an Information Resource Dictionary (IRD). This concept can be employed in database management to facilitate managing multiple databases. An IRD should be designed and developed to describe and control the information which is merged and consolidated from the system catalogs of each individual database. This application can be referred to as an IRD for DBA support of multiple databases. The IRD should also describe and control the DBA tools that operate on the enterprise data dictionary.

With the help of an IRD and a toolset, the DBA can describe, control, document, maintain, manage and track the information in the dictionary that describes the databases within the enterprise network. The IRD also allows the DBA to integrate information from various DBMSs; to enable the DBA to easily acquire information about different databases; to clearly identify the various types of databases available; to

present information in a succinct form to enable the DBA to quickly isolate where to look for information from the complex DBMS environment; to define, describe and control the DBA tools that operate on the IRD and on the databases within the network; and to permit extensibility to the IRD and the DBA toolset.

A consolidated data dictionary is required to help the DBAs deal with multiple databases. Ideally this consolidated data dictionary should contain at a minimum all information available in the individual system catalogs. However this may involve much redundant information. In this study, we confine ourselves to examining a portion of catalog information from different catalogs of multiple databases of importance to the DBA function. The data flow between the system catalogs and the consolidated dictionary is illustrated in Figure 3.13

Figure 3.13 Data Flow among System Catalogs and IRD

3.4.2 DBA Tools for Multiple Databases

A consolidated dictionary is useful only if the DBA can get at it and use it. This is true of any collection of any kind of data. So DBA tools are required to access, analyze, and report on the collected data. These tools access the consolidated dictionary and thus permit cross-analysis among different databases.

This study examines some of the repetitive functions that the DBAs have to perform at different databases. DBA tools are then evaluated to automate these functions so that the tools can operate on all the data collected from different databases as well as sending command operations back to different database. Most of these functions are analysis, monitoring, and reporting functions. Some of these functions which are identified in Section 2.1, may include:

- analysis of space availability and space usage
- monitoring the currency of enrolled users
- reporting on security information
- reporting on data fragmentation and chaining
- monitoring and reporting on database files that are almost full
- reporting on applications information
- reporting on database information
- performance of functions to enroll, delete and alter database users.

CHAPTER 4

A PROPOSED DBA TOOLSET ARCHITECTURE

One of the major objectives of this study is to describe the characteristics and architecture of a toolset for Database Administrators which would facilitate the management and administration of multiple databases. The proposed toolset architecture would fulfill the following objectives:

- to support the activities surrounding the management and administration of multiple databases;
- to capture and store information about all databases in use within an organization;
- to capture and store information that is required to support the administration of multiple databases;
- to improve the ability to monitor and control resource usage associated with each of the organization's databases;
- to provide a menu-driven or windows-based interface to the DBA Toolset;
- to support, as the repository architecture, the IRDS implementation;
- to improve the ability to recognize existing information resources related to databases.

In this Chapter, we discuss the IRDS Standards and describe why the ANSI IRDS was chosen for this study. We then examine the characteristics and functionality of the proposed DBA Toolset and the architecture upon which it would be constructed.

4.1 The IRDS Standards and Its Role

An Information Resource Dictionary System (IRDS) is a system which addresses the storage, control and retrieval of metadata (data about data). The American National Standards Institute (ANSI) has published a standard (X3.H4) which describes an implementation of such an IRDS. The International Standards Organization (ISO) also adopted an IRDS Framework in 1989. The IRDS represents an effort to provide data dictionary support for information management.

As organizations begin to address effective information management practices, they will adopt or evolve policies and procedures designed to formalize Information Resource Management (IRM) functions within the organization. Effective IRM policies can help coordinate the development, operation and maintenance of an organization's information systems and data.

Historically, the data processing function within an organization has had, as its priority, activities associated with the capture and transformation of data. The data processing function has emphasized these activities through the design and implementation of programs and processes which transform data rather than those activities which manage data. Although many organizations are now keenly aware of the need to manage data as an important (and often vital) resource, few organizations

have actually implemented the practices and procedures associated with effective IRM.

The concept of IRM derives from an understanding of the need to manage and utilize data as a primary resource within an organization. Interestingly, some organizations have renamed their Data Processing function with titles which give visible recognition to the increasing importance of data and information to the organization. Thus, we see names such as Information Resource Management Systems or Management Information Systems or Information Technology Systems becoming more frequent in their use.

The Information Resource Dictionary (IRD), as described in Appendix B and C, provides a focal point for the storage, coordination, processing, conversion and communication of an organization's data resource.

The IRD, with its extensibility mechanisms, can be used for purposes other than the management of primitive or derived data. It also has application in the areas of systems development, database administration, configuration management, operations and project management, to name but a few potential uses

The introduction of an IRDS within an organization can, in addition to providing an effective vehicle for managing the data resource, reveal some dramatic cost-savings to the organization. For example, a

preliminary cost-benefit overview prepared by the Institute for Computer Science and Technology (ICST) estimates that the United States Federal Government could realize over \$120 million in benefits by the early 1990's through the implementation of a standard IRDS. Opportunities identified in the ICST study for cost reductions and cost avoidance included the following [GOLD 87]:

- Improve the ability to recognize existing information resources which have a potential for being shared amongst those within the same organization;
- Simplify software and data conversion through the provision of consistent documentation;
- Increase portability of acquired skills resulting in reduced personnel training costs;
- Aid development, modification, and maintenance of manual and automated systems throughout their life-cycle;
- Support an organization-defined data element standardization program;
- Support records, reports and forms management, encompassing non-automated to fully automated elements.

The utilization of an IRDS may easily be extended to include the management and administration of a database environment. Such use of

an IRDS may also result in the realization of many of the same benefits as described above. In the next two sections, a short overview of the ANSI and ISO's IRDS structure is discussed. For further information, refer to the Appendix B and C and references [ANSI 88], [ISO 89a] and [ISO 89b].

4.1.1 ANSI IRDS - Overview

The ANSI IRDS has a modular structure. It specifies a core module (module 1) that contains the main functionality needed by organizations. The standard also includes a collection of optional modules: a basic functional schema (module 2), IRDS security module (module 3), extensible life-cycle phase facility (module 4), IRDS procedures facilities (module 5), application program interface facility (module 6), and entity lists facility (module 7).

The core IRDS module contains a menu-driven panel interface and a command-language interface. For additional flexibility, the module specifies capabilities that permit organizations to customize and extend the types of information that can be stored in the IRD.

The ANSI IRDS standard does not dictate a specific implementation approach. The IRDS could, for example, be implemented using a relational, network or object-oriented DBMS.

The ANSI IRDS architecture can be viewed as a four-layer architecture in which the information specified at one level describes (and potentially controls) the information stored at the next lower level. Figure 4.1 illustrates the four levels. Thus one level defines the types of "objects" which can be described at the next lower level, and that level

contains the "instances" of these objects. These four levels are briefly described below:

Level	<u>Layer Description in ANSI Term</u>
4	IRD Schema Description Layer
3	IRD Schema Layer
2	IRD Data Layer
1	Production Data Layer

Figure 4.1 Overview of ANSI IRDS Architecture

Level 4 is established by the ANSI IRDS standard committee which specifies the level 4 model. Level 3 is the fundamental data model used to design level 2. This level is populated and maintained by the corporate dictionary administrator. Level 2 describes application systems, processes, hardware, users, etc. Level 2 is the metadata and is primarily the concern of application developer, system analysts, DBAs, etc. Level 1, the lowest level, is end-user data.

The ANSI IRDS data model at level 4 is based on the Entity-Relationship-Attribute nomenclature. ANSI allows binary relationships only; relationships of degree three or higher must be modeled as entities. Relationships relate entities only, not other relationships. Relationship may have attributes.

The ANSI IRDS Standard Committee also specifies edit, control, and updating functions at level 3 and 2. These functions are provided by the Procedure Facilities which is described in Appendix B and in more detail in [ANSI 88].

ANSI also provides a small starter set of Entities, Relationships, and Attributes at level 3. This starter set is briefly discussed in Appendix B and more detail in [ANSI 88].

4.1.2 ISO IRDS - Overview

The ISO IRDS is evolving at this stage. The ISO IRDS 'framework' document [ISO 89b] has been adopted but the service interfaces are several years away [WINS 89a].

ISO IRDS also has four level of data as show below and these are also illustrates in Figure 4.1A

<u>Level</u>	<u>ISO Term</u>
4	Fundamental Level
3	IRD Definition Level
2	IRD Level
1	Application Level

The purpose of the Fundamental Level is to prescribe the types of objects about which data may be recorded on the IRD Definition Level. The purpose of the IRD Definition Level is to contain IRD definitions. The purpose of the IRD Level is to contain IRDs. The Application Level is the level at which instances of business data are recorded. For further details of each of the layer, refer to Appendix C or [ISO 89b].

Figure 4.1A Overview of ISO IRDS Architecture

The ISO data model at level 4 adopts the Object/Association approach. ISO disallows attributes on associations and does not allow many-to-many associations. Thus the ISO association is therefore, similar to a foreign key in the relational model.

The ISO Level 3 starter set is based on SQL as the ISO data model resembles the relational model.

ISO also provides IRDS Services Interface at two levels (level pair): the IRD Definition Level and the IRD Level. A level pair consists of two adjacent data levels. The successful operation of an IRDS requires the data on the lower level of any level pair to be consistent with a specified version of its schema at the higher level of the level pair. If a schema is changed, for which data exists at the lower level, consistency must be maintained between that data and the schema.

ISO is in the process of refining the IRDS Facilities and IRDS Interfaces standards. The purposes of these are the same as the ANSI. The ISO Services Interfaces proposal is also based on SQL. For more details regarding the ISO Facilities and Interfaces, refer to Appendix C and [ISO 89b].

4 1.3 Discussion

This section provides a brief summary of the differences between the ISO and ANSI standards. The terminology, relationship rules and interfaces are briefly compared below.

Terminology

ANSI and ISO each use different terms to describe the layers of the IRDS and the data model itself.

<u>LEVEL</u>	<u>ANSI</u>	<u>ISO</u>
4	IRD Schema Description	Fundamental Level
3	IRD Schema	IRD Definition Level
2	IRD Data	IRD Level
1	Production Data	Application Level

The terminology used for the data model differs in that ANSI IRDS uses "entity-relationship" and the ISO IRDS uses "object-association".

Relationship Rules:

The ANSI IRDS adopted an approach which permits only binary relationships but allows attributes to be associated with those relationships. Relationships with a degree of three or higher must be modelled as entities and binary relationships that relate entities only, not other relationships [WINS 89b].

The ISO IRDS does not allow attributes on relationships nor many-to-many relationships.

Interfaces:

The ANSI IRDS has developed a Panel Interface and a Command Language Interface, while the ISO IRDS focused initially upon a Service Interface. It is important to note that the ISO Service Interface is based on SQL, something that is missing from the ANSI IRDS.

For purposes of this study, the ANSI IRDS was chosen for reasons outlined below:

- it is an available, approved standard;
- others have implemented prototypes of the ANSI IRDS;
- more availability of expertise and experience;
- more reference material on the subject is available.
- some of the upcoming repository systems from the DBMS vendors are based on ANSI IRDS.

4.2 DBA Toolset Architecture - Overview

The architecture of the processes and data which will support the effective management of multiple databases consists of two sets of components -- components which address the storage of data about the multiple databases and their related resources, and components which address the capturing, accessing, monitoring, reporting, updating and control of that data.

In its simplest terms, the architecture may be depicted as follows:

The Processes components (the **Software Architecture**) contain the functions and commands to permit the DBA to monitor the status of

every database installed on the organization's computer network. The IRD components (the **Data Architecture**) encompass the data repository and the structures associated with that repository in which information is held about all the organization's databases and related resources.

We will refer this proposed system as Multiple Database Administration System (MDAS).

Each of these architectures is described below.

4.2.1 Data Architecture

The Data Architecture of the DBA Toolset is base on a subset of the ANSI IRDS standard. The Data Architecture is intended to provide a formal repository for the storage and control of metadata about the organizations databases. Its structure must also lend itself to modifying the types of metadata stored and to defining new types of metadata to be stored.

The ANSI IRDS implementation fully addresses two key objectives of the Data Architecture of the DBA Toolset, namely, the "formal" aspect of the repository and the "extensibility" aspect of that repository.

Because the repository of the DBA Toolset is to contain metadata about all the organizations databases which has been captured, consolidated and stored in the repository, the integrity, timeliness, and accuracy of that data is crucial to the effective operation of the DBA Toolset. Thus, there must be a formal, rigorous method of defining what metadata is to be stored in the repository as well as a definition of the means by which that data is to be captured and consolidated. This formality and rigor of definition becomes especially important when we consider the potential frequency at which a growing organization will add new databases to its computing network.

A repository which is extensible is also of high importance to the effective administration and control of the organizations data resource. As the organizations information resource management needs evolve and change over time, the extensibility mechanisms of the repository can be used to reflect these needs. This permits the DBA to "tune" the repository such that new types of data about databases are captured and stored. It also affords the DBA the facility to accommodate situations where the organization implements a new database type (from a different database vendor) or a new type of computing device is attached to the network. In each case, the DBA need merely add these new definitions into the repository and is thus able to respond to these types of situations in a timely and efficient manner.

The data architecture of the DBA Toolset employs the concept of layers as described by the ANSI IRDS. The architecture addresses three such layers: IRD Schema Description Layer, IRD Schema Layer and IRD Data Layer.

The Data Architecture of the DBA Toolset is shown in Figure 4.2. As discussed in Section 4.1.1 and in Appendix B, the IRD Schema Description Layer is fixed by the ANSI IRDS standard committee, thus this study will not address this layer. Moreover, the Production Data Layer is end-user data and IRDS does not address this layer. This

study focuses on the design and definition of the IRD Schema Layer and IRD Data Layer. The design and definition of each of these two layers is discussed in more detail in Section 4.3.1.

As noted at the beginning of this section, the Data Architecture of the DBA Toolset is based on a subset of the ANSI IRDS. It was not the objective of this study to develop a full implementation of all 7 modules of the IRDS, but rather to explore the concept of implementing a formal, extensible repository in which to store metadata about multiple databases in use within an organization.

Figure 4.2 Data Architecture for the DBA Toolset

4.2.2 Software Architecture

The Software Architecture represents the structure of the process components which interfaces with the Data Architecture. There are six software components in the DBA Toolset -- the MDAS Interface, the Catalog Extraction Programs, the Command Programs, the Access Programs, the IRD Operation Programs and the DBA Toolset Administration component. An overview of the components of the Software Architecture may be depicted as follows:

- **MDAS Interface**

This component contains the processes which permit the DBA to execute any and all of the functions (or capabilities) of the DBA Toolset. The DBA Interface is the only means through which the contents of the repository may be accessed. This is an important consideration toward ensuring the integrity of the IRD. The toolset could support these interface types: panel interface; command interface and service interface. These will be discussed in Section 4.3.2.

- **Catalog Extraction Programs**

This component provides the capability to extract data from each database in use within the organization and transfer it to the IRD. These programs are run in batch processes (Catalog Extraction

Processes) which operate against remote databases at times and frequencies established by the DBA.

- Access Programs

This component contains the administrative, analysis and management "tools" which permit the DBA to query, report and analyze the data in the IRD

- Command Programs

This component provides the capability to operate on any remote database by sending database commands as batch process (Remote Database Operation Process).

- IRD Operation Programs

This component provides the administrator of the the repository with the ability to query, report, modify and/or extend the definitions of the data stored in the IRD.

- MDAS Administration Programs

This component provides the "housekeeping" facilities to the administrator of the DBA Toolset. It provides for example the ability to register new users as being authorized to access and use the Toolset.

Figure 4.3 depicts the components of the MDAS Architecture.

Figure 4.3 MDAS Architecture

Since MDAS is designed for a multiple database environment in which databases are not linked, a mechanism is required for MDAS to communicate with the remote databases. This mechanism can be either a remote procedure call or a batch process submitted for execution to the remote nodes. In MDAS, the Catalog Extraction Process and the Remote Database Operation Process are both implemented using remotely submitted batch processes. This approach is taken because this facility was available as an integral component of the underlying operating system, and remote procedure calls are not currently supported by the operating system.

In a distributed database environment, one alternative architecture for MDAS is as following. MDAS is installed on one of the databases, therefore Catalog Extraction Process and Remote Database Operation Process can communicate with the other connected databases. This approach permits MDAS to operate in an interactive as well as batch mode, for sending database commands or extracting system catalog information. In the following, issues are described that one should take into consideration when applying the concept of MDAS to distributed database environments:

- 1) MDAS operates on system catalog data extracted from various local databases. In a distributed database environment, system catalog

extraction can be performed in an interactive mode as well as batch mode. If extracted interactively, an improvement in currency of meta data is achieved as compared to that obtained using the batch mode. This is possible when system catalogs of the distributed DBMS can be accessed interactively using an extraction program.

2) It is important to note that the scope of this thesis excludes performance tools. Although many performance tools exist for centralized databases, no comprehensive performance tool for distributed databases exist to date. This is due to some non-trivial issues such as optimization of global database, data fragmentation and allocation, query optimization, etc. exist that require further treatment. Such problem does not exist in a multiple database environment because optimization is done locally.

In order for this study to fully explore the design of MDAS for one type of environment, this thesis confines the scope to the multiple database environment.

4.3 DBA Toolset Components - Design Description

Introduction

The key component of the DBA Toolset is the repository of database metadata. This repository is the central storage point for information about all databases in use within the organization. These databases may be of differing types (eg. ORACLE, Rdb, INGRES) and they may reside on different computers (ie. nodes on the organization's computer network).

The actual methods (command strings) necessary to access the databases and the location of the catalog extraction programs are stored in the IRD. The Catalog Extraction Programs access this information in the IRD in order to "tailor" the extraction process to a particular node and database. The Catalog Extraction Programs then are submitted as batch processes to the target nodes and, when executing, access the target databases according to the method defined in the IRD. Each such running Catalog Extraction Program extracts the requested data from the system catalogs and then populate the IRD with metadata extracted from each of the system catalogs.

The Command Programs are the means by which the DBA can perform administrative operations on any of the organization's databases, such as adding a user to the system or granting a privilege to a user. The Command Programs access the target databases using the command

stored in the IRD, utilizing remotely submitted batch jobs in a manner similar to that used for the Catalog Extraction Program.

The balance of this section is devoted to a design description of each of the components of the proposed DBA Toolset. A prototype, illustrating the practical application of these concepts, is described in Chapter 5.

4.3.1 The Information Resource Dictionary

An Information Resource Dictionary (IRD) is a repository which contains information about an organization's metadata. An IRD for an organization is composed of many elements of data which are used in support of various information management functions that the organization wishes to exercise. An IRD may be used to support a variety of information management functions such as Data Administration, Systems Development, Document Management, Network Administration, etc. Canadian government departments increasingly are implementing IRD's to serve their information management needs. This study explores a particular application of an IRD; to facilitate the functions of Database Administration.

As described in Chapter 2, the database administration function defines, maintains and controls the definition, design and performance of the physical databases in use within an organization. The database administration function has, as its highest priority, the responsibility to ensure the availability, performance and the integrity of the organization's databases.

The application of an IRD to the database administration function is the focus of this study. A centralized IRD can provide the DBA with a repository in which information about the organization's databases can be

collected and stored. The IRD then becomes the one repository where the status of any and all databases in use within the organization can be monitored and controlled. Through the use of a standardized repository, a degree of hardware platform and database independence is realized. This permits the DBA to monitor the status of any type of relational database operating on any type of hardware platform.

The IRD examined by this study is intended to assist the DBA in managing multiple databases in use within an organization. We now describe the characteristics of such an IRD.

The Information Resource Dictionary contained within MDAS is itself a specific use of relational database technology. The IRD has a database schema which describes the types, relationships and attributes of the data to be stored. Some of the data stored in the IRD are elements extracted from the system catalogs of all the databases in use within the organization. A relational implementation of the IRD was chosen, because it most closely matched the available technology within the target platform. However, any other implementation such as a network implementation or an object-oriented implementation of the IRD could also have been adopted.

Considering first the specifics of the metadata contained in the IRD that we wish to be able to monitor on the status of each database as a

whole. For example, we need to know the database name, its location, its type, its status (available/not available), its hardware platform, etc. For each specific database, we need to know about such items as resource usage, fragmentation, authorized users, backup/recovery status and history, etc.

Since some of the metadata stored in the IRD represents the results of extraction and consolidation of metadata contained in the system catalogs of multiple databases, provision must be made to ensure that the metadata is as current as possible. The mechanics of this process is described in Section 4.3.3.

The schema of the IRD is represented in terms of Entity Types, Relationship Types and Attribute Types which are used to describe the data contained in the IRD. Detailed descriptions of these metadata types are provided in [GOLD 87], [ANSI 88] and [WERT 89]. The most significant feature of the ANSI specification for the IRD is its use of the data model abstraction as the means of specifying the structure and content of the IRD. This approach provides for self-description of the IRD and permits the schema to be extended to service a variety of information resource management needs [PRAB 90a].

Using this approach to an IRD, the IRD administrator can define and describe the types of database information to be stored in terms

which are specific to the needs of the organization. The DBA is not constrained by any vendor-specific implementation of the dictionary or by the vendor-specific implementations of the RDBMS system catalogs.

Further, this ability to describe the contents of the dictionary in terms specific to the information resource management needs of the organization is not limited to the initial implementation and population of the IRD.

As the organization's needs evolve over time, the structure and content of the IRD may easily be altered to reflect these changed needs.

For an organization such as AECL Research, whose information processing is conducted at 3 sites, the IRD would likely be implemented at each site. This would provide each of the local site DBAs with the means to monitor the status of the multiple databases for which they are responsible. Database links (using products such as ORACLE's SQL*NET) could be established to link the IRDs at 3 sites and these would permit the global DBA to monitor the status of all databases in use within the organization.

The design of the IRD structure for the DBA Toolset embodies the concept of layers as per the ANSI IRDS implementation. Each of the repository layers of the DBA Toolset's IRD is described in the next two sections.

4.3.1.1 IRD Schema Layer

Residing at the second highest level of the DBA Toolset data architecture, the IRD Schema Layer provides for the definition of the entity types, relationship types and attribute types which may be stored in the IRD Data Layer. The Schema Layer specifies the objects to be stored in and controlled by the IRD Data Layer in terms of meta-entities, meta-relationships and meta-attributes.

Certain of the meta-entities contained in the ANSI IRDS starter set are appropriate for inclusion in the IRD of the DBA Toolset. Table 4.1 gives the sample of the ANSI IRDS Starter Set [WINS 89].

Table 4.1 ANSI Level 3 Starter Set

ENTITY TYPES	RELATIONSHIP CLASS TYPES	ATTRIBUTE TYPES (OF ENTITIES)	ATTRIBUTE TYPES (OF RELATIONSHIPS)
SYSTEM	CONTAINS	ACCESS-NAME	RELATIVE-POSITION
PROGRAM	PROCESSES	ADDED-BY	FREQUENCY
MODULE	RESPONSIBLE-FOR	ALLOWABLE-VALUE	
USER	RUNS	ALTERNATE-NAME	
FILE	GOES-TO	CLASSIFICATION	
RECORD	CALLS	CODE-LIST-LOCATION	
ELEMENT	DERIVED-FROM	COMMENTS	
DOCUMENT		DATA-CLASS	
		DATE-ADDED	
		DESCRIPTION	
		DESCRIPTIVE-NAME	
		DOCUMENT-CATEGORY	
		DURATION-TYPE	
		DURATION-VALUE	
		HIGH-OF-RANGE	
		LAST-MODIFICATION-DATE	
		LAST-MODIFIED-BY	
		LOCATION	
		LOW-OF-RANGE	
		NUMBER-OF-LINES-OF-CODE	
		NUMBER-OF-MODIFICATIONS	
		NUMBER-OF-RECORDS	
		RECORD-CATEGORY	
		SECURITY	

It is necessary however to define and describe other entity, relationship and attribute types which reflect the information stored in organization's databases. This would include for example entity types which describe the definitions of Database, DBMS, Hardware, etc. In Section 3.2, we discussed issues such as heterogeneous database and hardware environments; proliferation of databases; increasing data complexity; etc. Let us now briefly discuss how the DBA Toolset handles the information related to these issues using the extensibility of IRDS. For example, new entity-types such as OBJECT_BASE, KNOWLEDGE_BASE, etc. can be added in addition to DATABASE. This allows us to capture different classes of information. Moreover, we can add GRAPHIC-STRING and RULES to capture the new types of data to be used by the organization.

Figure 4.4 [CSR 88a] shows that the IRD Data Layer control information forms only part of the full IRD Schema Layer. Other metadata (not addressed in this study) could be captured within the IRD Schema Layer including repository control information such as security and life cycle control. Schemas which are under development but not yet "active" may also reside in the IRD Schema Layer as can old schemas that have been replaced and are now "inactive". More

Information on this area can be found in [GOLD 87], [GOLD 88d], and [ANSI 88].

Figure 4.4 Types of Information at the IRD Schema Layer

4.3.1.2 IRD Data Layer

The purpose of the IRD Data Layer is to contain the Information Resource Dictionary (IRD). This is the layer on which schema descriptions from the system catalogs of each database are stored. Moreover metadata about Production Data Layer's entities, attributes and relationships will be recorded in the IRD Data Layer. Corresponding instances are on the Production Data Layer. Application source programs refer to the data types which are specified at the IRD Data Layer, when such programs are executed, then data instances on the Production Data Layer are retrieved and possibly updated.

Not all of the content of the IRD defines types of information at the Production Data Layer, as illustrated by the Venn diagram [CSR 88a] in Figure 4.5.

Figure 4.5 Types of Information at the IRD Data Layer

For example, in addition to containing the information that DATABASE Record is a record type and DBMS Type is a data element within that record type, the IRD might also contain information about which programs use particular record types or data elements (i.e. describing processes rather than defining data types). Other information in the IRD may help DBAs carry out activities (such as database naming standardization) for which they are responsible.

The purpose of meta-data in the IRD is to enable the IRDS to support the design, construction and operation of computer information systems, and any other functions for which the IRDS may be an appropriate tool. Most metadata in the IRD Data Layer of the DBA Toolset will be placed there by the IRD Administrator and the DBA.

The IRD Data Layer will also contain information about the organization's database environment, such as the purpose of each registered database, computer where the database resides, and processes which occur at the Production Data Layer.

As the database environment evolves towards an ever changing environment, this metadata will actually control handling of data at the Production Data Layer and may be used directly by database management systems or other software.

The IRDS standards make no restriction on the types of metadata which an organization may choose to hold in the IRD, provided that the instances of this meta-data correspond to the types which have been defined in the schema at the IRD Schema Layer. For example, new DBMS types and DATABASEs could easily be added to the DBMS and DATABASE Entity Types to accommodate the new operating environment, provided the entity types DBMS and DATABASE have been defined in the schema at the IRD Schema Layer.

4.3.2 The MDAS Interface

ANSI IRDS has defined three categories of interface which are discussed in Appendix B and in more detail in [ANSI 88]. However, the current design of MDAS interfaces do not conform to the ANSI IRDS standard for interfaces.

The MDAS Interface is the means (and the only means) by which the DBA accesses the contents of the IRD and by which the functions contained within the DBA Toolset are executed. Because the integrity and timeliness of the data contained in the IRD is so critical to effective administration of multiple databases, access to the IRD must only be through the MDAS Interface and that access must be carefully controlled through the implementation of security procedures built into the MDAS Interface. The MDAS Interface affords controlled access to the following:

- Catalog Extraction Programs;
- Command Programs;
- Access Programs;
- IRD Operation Programs
- MDAS Administration Programs.

Ideally, the toolset should support 3 interface types:

- A Panel Interface by which an interactive user can utilize the MDAS tool;
- A Command Interface for interactive use as well as for utilizing the MDAS from command files;
- A Service Interface that allows the MDAS to be called directly from other software for the purpose of utilizing it in embedded systems.

This study focuses on the design of the panel interface (MDAS Interface). The MDAS Interface is a menu driven/windows-based series of screens which permit the DBA to execute all of the functions and capabilities of the DBA Toolset. The MDAS Interface would also contain an online, context-sensitive help module with which to provide assistance to new authorized users of the DBA Toolset. The MDAS Interface should also allow easy addition of new functionalities to the toolset. This would overcome one of the limitations (could not extend the functionalities) of the surveyed DBA tools discussed in Chapter 2.

A conceptual diagram of the MDAS Interface is illustrated in Figure 4.6. A prototype of the MDAS Interface has been designed which incorporates some of the concepts described above. It is detailed in Chapter 5.

Figure 4.6 Examples of MDAS Interface

4.3.3 The Catalog Extraction Programs

The Catalog Extraction Programs (CEP) represent the means by which metadata of interest to the DBA is transferred from the system catalogs of each of the multiple databases and placed into the IRD.

There are two possible ways to upload the information into the IRD from the system catalogs of multiple databases. One way is to upload the information in a regular fashion such as nightly, hourly, etc. or more irregularly based on when information when local changes are detected in the individual databases. This requires autonomous processes at the local database that determine when uploading is necessary and that perform the upload at the required time. The second method is to upload the information on demand from a central location, by effectively "polling" the databases to obtain the information.

In the first case, a set of detection programs (triggers, rules, etc.) are required at the individual nodes so that any local operation that reaches a certain threshold value will trigger the upload of information to the IRD. This approach requires that the catalog extraction programs and their triggering mechanisms be resident at each remote node. Thus the CEPs can be invoked when the upload process is initiated by any one of the detection programs. The DBA controls the upload process by setting the various threshold values in the remote databases. The

information captured by this mechanism is more current due to its automatic uploads when conditions change.

In the second case, the MDAS generates a Batch Job File (BJF) and sends it over to the target node to execute against the remote database and upload the extracted data into the IRD. During the generation of the BJF, the required CEPs are appended to the BJF. Thus only one copy of each CEPs is required at the MDAS node and there is no need to maintain duplicated copies of the CEPs at the remote nodes. The only file that is located in the remote node is the BJF which is waiting in the batch job queue. This batch job is submitted to the remote node according to a frequency (nightly, hourly, etc.) specified by the DBA. Thus the frequency of uploading can be easily adjusted by the DBA to meet the requirements and importance of individual databases.

In this study, the second approach was chosen since it was simpler and easier to implement and maintain, since it only requires one and only one copy of each CEP at the MDAS node and no duplicated copies are required at the remote nodes. Also there is no need to develop and maintain detection programs at the remote nodes, or mechanisms that ensure that the autonomous processes at the remote nodes stay alive and functioning. Finally, the second approach does not impose extra

overheads to the DBMS kernel since there is no need to detect and react to any changes in the DBMS kernel's catalog.

Let us now look at the design of the catalog extraction programs. The characteristics of the catalog extraction programs are such that an organization will have many different types of CEP's operating at different time frequencies, on different vendor's databases and on different hardware platforms. It is therefore fundamental and key that the Catalog Extraction Programs be implemented in a flexible, customizable manner. Again, the IRD is used to describe and control the Catalog Extraction Programs. The IRD then not only contains the metadata needed by the DBA to monitor the status of the organization's databases, it also contains the information necessary to describe and control the execution of the Catalog Extraction Programs. This provides the DBA with considerable flexibility and control over the characteristics of the Catalog Extraction Programs. It permits, for example, the DBA to change the frequency attribute of a particular Catalog Extraction Program which in turn would alter the operational frequency of all occurrences of that CEP throughout the organization. It would also permit the DBA to add new CEP's into the DBA Toolset without having to associate physical copies of the CEP with specific databases.

Another ideal design characteristic of the CEP's requires that they accept run-time parameters so that one CEP may be executed against any or all databases of the same type. In addition, the CEP should derive as many of these parameters as possible via the IRD in an interpretive manner. The DBA, through a screen interface, would identify which database, or databases, a particular CEP was to act upon and its time and frequency of execution. The CEP would use the database identifier to retrieve node and access string information via the IRD just prior to its execution. Implementation of this design characteristic permits the CEP to execute against similar database types which may be in use on computer nodes provides by different vendors (e.g. DEC VAX, SUN, Hewlett Packard).

Some examples of the types of CEP's are as follows:

CEP1 : Executes against Production Databases at 15 minute intervals. It captures information related to availability of the databases and available space in certain tablespaces and tables and transfers this information to the IRD. The DBA can increase or decrease the frequency at which CEP1 executes depending on the criticality of the database to the organization.

CEP2 : Executes against Test and Development Databases and operates on a nightly basis. This type of program would capture and

consolidate information regarding the status and usage of all aspects of these databases and transfer the resulting data to the IRD. Since these databases are usually not considered to be as critical to the organization, it would be appropriate to capture this information only after the close of business each day.

In each of the examples above, the DBA controls the actions of the CEP by describing the CEP's attributes to the IRD. A conceptual diagram of the Catalog Extraction Program is illustrated in Figure 4.7.A prototype CEP has been designed which incorporates some of the concepts described above. It is detailed in Chapter 5.

Figure 4.7 The Concept of Catalog Extraction Process

4.3.4 The Access Programs

The Access Programs (AP) represent the means by which the DBA may produce reports, execute queries and analyze the data contained in the IRD. The Access Programs may be thought of as the set of functions which allow the DBA to effectively monitor and analyze the status of any or all databases in use within an organizations. The Access Programs perform monitoring and analytical functions such as:

- report on space availability within a database;
- analysis of space usage;
- analysis of table fragmentation;
- report on application objects residing in a database;
- report on tables used by applications in database;
- report on authorized users for a database;
- report on user quotas;
- report on access privileges.

In each of the above cases, the report or analysis could be directed at the status of a particular database, or be for all databases in use within the organization.

Since all metadata being queried by the Access Programs resides in IRD, two significant benefits are realized. The first is that there needs to be only one set of Access Programs to perform analysis and reporting

on the status of any type of database on any hardware platform in use within the organization. Thus the DBA may tailor the design and implementation of the Access Programs to reflect exactly the information she or he needs to monitor and analyze the status of the organization's databases. This can be done in the secure knowledge that the Access Programs will work for any type of relational database operating on any type of hardware. This overcomes one of the limitations of the surveyed DBA tools.

The second benefit occurs from the ease and efficiency with which the DBA can monitor and analyze multiple databases. Since all the necessary data is stored in the IRD, the DBA can perform monitoring and analysis of the databases at a frequency and with a thoroughness which would not have been previously possible.

A conceptual diagram of the Access Program is illustrated in Figure 4.8. A prototype of the Access Programs has been designed which incorporates some of the concepts described above. It is detailed in Chapter 5.

Query On Fragmented Tables					
DB ID	Table Name	Owner	Total Allocation	Used Bytes	% of Fragments
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____

Enter Your Query Criteria, then press <Execute Query>

Figure 4.8 Sample Access Program

4.3.5 The Command Programs

The Command Programs (CP) represent the means by which the DBA can perform administrative operations on specific databases, without the need to physically sign on the specific databases and nodes.

For example, Command Programs may be used to enroll new users, revoke users, change access privileges, enable/disable auditing, create public views and synonyms, etc. Each of these functions would be executed against a specific target database.

As was the case with the Catalog Extraction Programs, the Command Programs must be implemented in such a way as to provide the greatest degree of independence over what database is administered, where it is located, what type it is and upon which hardware platform it operates. And again, the Information Resource Dictionary may be used to describe and control the various types of Command Programs.

The Command Programs may be thought of as a series of discrete functions. The sum of these functions (CP's), represent the day to day administrative operations performed by a DBA on specific databases. Only one Command Program is required for a specified database operation on the same database type.

For example, CP1 would add users to an ORACLE database residing on a Sun Workstation and CP2 would add users to a DEC Rdb

database residing on a VAX. Also CP1 can operate on any ORACLE databases while CP2 can operate on any Rdb databases.

If for example, the DBA wished to add a user to a particular database, he would select the Add User option in the MDAS Interface (described in Section 4.3.2). This would present him with a panel which would in turn prompt him to identify the user, the user's access privileges and the database into which the user is to be enrolled. Once that information was provided, the Command Program associated with the Add User option would then execute the required commands to enroll that user in that database. The particular Command Program which executed would be specific to the type of database and hardware platform on which the database resides.

Other functions can easily be added to the toolset to allow the DBA to perform any database operation at the remote database(s). Since databases are not linked, MDAS generates a Batch Job File (BJF) and sends it over to the target node to execute the remote database. During the generation of the BJF, the required command program is appended to the BJF. Thus only one copy of each CPs is required at the MDAS node and there is no need to maintain duplicated copies of the CPs at the remote nodes.

A conceptual diagram of the Command Programs is illustrated in Figure 4.9.

Thus the IRD node can be considered as the command centre where database administrative functions can be sent concurrently to many remote databases. This design overcomes one of the limitations (services one database at a time) of the surveyed DBA tools in Chapter 2.

A prototype CP has been designed which incorporates some of the concepts described above. It is detailed in Chapter 5.

Figure 4.9 The Concept of Sending Commands to Target Databases

4.3.3 The IRD Operation Programs

The IRD Operations programs provide the administrator of the IRD with the ability to query, report, maintain and extend the data stored in the IRD. The IRD Operation programs may be thought of as the set of functions which allow the administrator to effectively maintain the data in the IRD Schema and IRD Data Layers. The IRD Operation programs allows the authorized users to perform functions such as:

- report on IRD Schema and Data Layers;
- query on IRD Schema and Data Layers;
- modify metadata in the IRD layers such as
 - modify an existing entity-type
 - update an existing software information
 - etc.
- add new metadata to the IRD layers such as
 - add new entity-type, attribute-type, etc.
 - register new database, hardware, DBMS, etc.
 - etc.
- delete metadata from the IRD layers such as
 - delete an entity-type
 - de-registration of database.

Only the IRD administrator has the privileges to alter, modify, add and extend the metadata stored in the IRD Schema Layer and the IRD Data Layer. The authorized DBA has the privilege to populate the IRD Data for which they are responsible. All other users such as project leaders, programmers, etc. only have query or report capabilities.

The IRD Operation programs contain menus, screens, and programs. The menus are used to control access; the screens allow the users to perform a query by entering querying criteria; the maintenance screens allow the administration to modify, delete or add metadata to the IRD; the programs generate reports on the metadata.

A set of prototype IRD Operation programs have been designed which incorporated some of the concepts described above. It is detailed in Chapter 5.

4.3.7 MDAS Administration

The MDAS Administration component provides the "housekeeping" facilities for the administrator of the DBA Toolset. This component allows the administrator to enroll, revoke and alter user access to the DBA Toolset. It also allows the administrator to report on the DBA Toolset user information.

This component consists of menus, screens and programs. The menus are used to control access to this component. The screens allow the administrator to add, revoke, and alter the user privileges. The programs are called by the screens to grant or revoke access privileges to or from the users.

A set of prototype Administration programs have been designed which incorporated some of the concepts described above. They are detailed in Chapter 5.

CHAPTER 5

DESIGN AND DEVELOPMENT

OF

A DBA TOOLSET PROTOTYPE

Introduction

In Chapter 3 we described the variety of tasks and challenges being faced by the Database Administration function in today's ever-changing computing environment. In Chapter 4 we described the concept of a DBA Toolset and the architecture upon which it could be based. In this Chapter we describe the design of a prototype which reflects solutions to the tasks and challenges faced by the DBA and is constructed according to the data and software architecture previously described.

In designing the DBA Toolset prototype, we employ the actual computing environment of AECL Research Company - Chalk River Laboratories (discussed in Section 3.3.3). This environment currently exhibits many of the characteristics which would benefit from the implementation of the proposed DBA Toolset. These are summarized as follows:

- multiple databases;
- multi-node computing network;
- moving towards heterogeneous database types;
- heterogeneous hardware platforms.

This prototype focuses on the database administration activities associated with 3 specific nodes on the Chalk River Laboratories computing network:

- CC1: is a Production node and contains 3 databases:
 - PROD (Oracle Ver 5.1.22)
 - PROD2 (Oracle Ver 5.1.22)
 - PROD6 (Oracle Ver 6.0.30)
- CM2: is a Development node and contains 2 databases:
 - DEV (Oracle Ver 5.1.22)
 - DEV6 (Oracle Ver 6.0.30)
- CM1: is a Development node and contains 2 databases:
 - DEV (Oracle Ver 5.1.22)
 - DEV6 (Oracle Ver 6.0.30)

Each of the computer nodes are DEC VAX's and operate the VMS (Ver 5.4) operating system.

The DBA Toolset prototype is implemented on node CM11 in the DEV6 database. This node is also referred to as the IRD node in this chapter. Although the computer nodes are linked through the organization's network (RCNET), there is no software in place which links any of the databases. This then is the multiple database environment described in Chapter 3.

To simulate the prototype operating on different database types (heterogeneous databases), the prototype will address the extraction of data from Oracle Version 5 and Oracle Version 6 system catalogs. Since the differences in Version 5 and version 6 are significant, especially in terms of the underlying system catalogs, one may appropriately view

them as different database types. The environment of the prototype is illustrated in Figure 5.1.

The prototype is implemented using the relational approach but it could also be implemented using a network or hierarchy approach if desired. However as mentioned in Chapter 1 that the scope of this study is confined to relational databases which dominate the DBMS market at present. Moreover, the prototype environment is all SQL engines, therefore it is natural to employ the relational approach for implementing the prototype as well.

Figure 5.1 The Prototype Environment

The development of the DBA Toolset prototype is accomplished by considering each of the seven components:

- identify DBA functions to be automated;
- develop the Information Resource Dictionary;
- develop the Catalog Extraction Programs;
- develop the Access Programs;
- develop the Command Programs;
- develop the IRD Operation Programs;
- develop the MDAS Administration Program;
- develop the MDAS Interface.

The development tools used to construct the prototype are those tools from Oracle Corporation which are integrated with Version 6 Oracle RDBMS. Specifically, the prototype uses:

- SQL*Forms V3.0;
- SQL*Menu V5.0;
- SQL*Plus V3.0.

SQL*Forms is a development tool which may be used to design interactive screens for entry of data and querying of database tables. It employs a 4th generation procedural language called PL/SQL which permits the developer to define routines (called triggers) which determine the logical events to occur upon the entry of data.

SQL*Menu is a product which is used to design menu options for each of the functions to be included in the DBA Toolset prototype. Each of the menu options is in turn linked to a SQL*Forms screen so that the DBA may enter parameters which further define the functions to be executed. SQL*Menu also has within it an access control component which is used to define and control access to all functions within the DBA Toolset prototype. This ensures that only authorized users (DBA's) are permitted to view and change the contents of the IRD.

SQL*Plus is Oracle Corporation's implementation of the Structured Query Language. Oracle has added some formatting, calculation and editing facilities to SQL resulting in SQL*Plus. For purposes of the DBA Toolset prototype, SQL*Plus is used primarily for the Catalog Extraction Programs and for the Command Programs.

The details of each component are described in the following sections.

5.1 Functions Automated by the Prototype

The Access Programs within the DBA Toolset operate on consolidated data which has been captured from the system catalogs of the organization's databases and placed in the IRD. By analyzing these consolidated data in the IRD, the DBA can gain a comprehensive understanding of the status of each of the organization's databases. The type of information which will be stored in the IRD depends then on the functions to be automated within the DBA Toolset. These different types of information constitute the entities for the IRD.

To identify the entities for the IRD in the DBA Toolset prototype, we first must determine which DBA functions would be the most appropriate and beneficial to be automated.

In Section 3.1 we discussed the major differences in the administration activities associated with a single centralized database (review single system catalog) and those associated with multiple databases (review multiple system catalogs). As Figure 5.2 illustrates, the number and frequency of repetitive actions performed by the DBA to monitor and control the status of multiple databases can easily reach the point where available time will not permit the effective monitoring of all the databases within the organization. When this point is reached, the DBA can no longer provide assurances to the management of the

organization as to the availability, performance and integrity of the organization's information resources.

Therefore, in considering which DBA functions to automate, those which are performed frequently and over a wide array of databases would be appropriate candidates to be included in the DBA Toolset prototype. Two functions which meet these criteria are the functions related to Access Control and Space Management.

These DBA functions are performed frequently (multiple times per day), and are aimed at virtually every individual database in use within the organization. These functions, and their associated entities are presented in Table 5.1. The individual databases which would be monitored and controlled are shown in Table 5.1A.

Figure 12 Typical Activities for the DBA at Oak Ridge Laboratories

Note: There are no database links amongst the databases.
The computers are linked by a local area network.

Table 5.1 Type of System Catalog Information

<u>DBA Functions to be Automated</u>	<u>Type of Information</u>
Analysis on space availability	Tablespace, Database File
Display % free of tablespace	Tablespace, Free Space
Analysis of space usage	Tablespace, Tables, Indexes
Report on account privileges & quota	Database Account, Tablespace
Report on possible fragmented tables	Table
Report on indexes of table	Index

Table 5.1A

Information to be Captured at/Command Sent to each Database

<u>Info to be Captured</u>	<u>Extracted From Node/Database</u>
Space availability and Free space information	CC1:PROD,PROD6; CM2:DEV6
Database Account with Resource or DBA privileges	CC1:PROD,PROD6; CM2:DEV6
Fragmented Tables	CC1:PROD,PROD6; CM2:DEV6
Indexes	CC1:PROD,PROD6; CM2:DEV6
Tablespace Information	CC1:PROD,PROD6; CM2:DEV6
Database File information	CC1:PROD,PROD6; CM2:DEV6
<u>Commands To Be Sent</u>	<u>Target Database</u>
Set up Database User Accounts	CC1:PROD,PROD2,PROD6

Other information to be captured by the prototype to help the DBA to manage multiple databases includes the following:

- databases residing in the company network;
- DBMSs available within the company;
- hardware connected to the company network;
- communication network;
- database backup;
- scheduled database shutdown operations;
- software tools;
- software tools installed in each database;
- application systems;
- application contacts;
- application programs;
- application database accounts.

This information augments that captured from the system catalogs of the individual databases and provides the DBAs with a clear overall picture of the databases and related resources residing in the company network. The entities related to this information is discussed in the next Section.

5.2 Toolset Components

The design of each of the DBA Toolset components are discussed in this section.

5.2.1 The Information Resource Dictionary

The prototype employs a relational database (ORACLE RDBMS) to implement the IRDS. The prototype is designed to satisfy two major requirements: it contains metadata that describes the logical structure and content of the IRD, and it contains metadata that describe the organization's individual database information resources.

The **IRD Schema Layer** contains three types of metadata for describing the organization's database information resources: Entity-Types, Attribute-Types, and Relationship-Types. Entity-types are the objects (types of information resources) about which we want to store data, such as DATABASE, SYSTEM, DBMS, HARDWARE. Attribute-types are the characteristics and properties that describe each entity-type, such as LENGTH, NAME, DATE-MODIFIED. Relationship-types are the connections between entity-types, such as DATABASE-CONTAINS-SOFTWARE, SYSTEM-USES-DBMS.

The **IRD Data Layer** contains the entities, attributes, and relationships that are occurrences of the corresponding entity-types, attribute-types, and relationship-types. These metadata describe individual

database information resources. For example, the entity `APPL_ID` is an occurrence of entity-type `ELEMENT`; `LEN` is an occurrence of attribute-type of `LENGTH`; `MDAS-USES-ORACLE` is an occurrence of relationship-type `SYSTEM-USES-DBMS`.

In the prototype database, the IRD Data Layer and the IRD Schema Layer are implemented as two separated sets of ORACLE tables to maintain the semantic distinction between types and occurrences of types.

The detailed designs for the prototype's IRD Data Layer and IRD Schema Layer are discussed in this section.

5.2.1.1 IRD Data Layer

The IRD Data Layer is the level on which metadata extracted from the system catalogs of each database are stored. Other metadata associated with the DBA functions is also stored at this level. Such metadata helps the DBA to run the administrative activities. As previously discussed in Section 4.3.1.2, the IRD Data Layer also contains information to describe and control the following types of information:

- Production Data Layer database schemas and definitions
- Information on databases, systems, programs, modules, processes, etc.
- Information on users and organizations
- Information on documents and reports.

Some of the information captured in this layer by the prototype are presented in the previous Section (5.1) and the entities required to describe the prototype's IRD Data Layer data are given in Appendix D.

The Entity-Relationship (E-R) model for the prototype's IRD data is illustrated in Figure D-1 in Appendix D. As well the schema descriptions and the detailed schema design for the IRD Data Layer are given in this Appendix.

5.2.1.2 IRD Schema Layer

A schema provides the means of representing information in a data dictionary. The construction of the schema determines the **types** of information that can be captured in the data dictionary.

As in any program whose variable types must be defined at the beginning of the program, **metadata types** must be defined before using a data dictionary. While IRDS provides a number of predefined metadata types in the Minimal and Basic Functional Schemas, the IRDS user can introduce additional metadata types if necessary, using the extension mechanism provided by the IRDS Core.

As data types must be specified appropriately in each program, metadata types must be specified appropriately in each IRD schema. The **IRD Schema** determines the types of information that the user can represent in the IRD.

Unlike many data dictionary systems, the IRDS provides a fully **extensible schema**. This schema extensibility gives users nearly complete flexibility in representation. The benefit of this extensibility also imposes a burden, however, in that the user must be responsible for the structure of the schema.

A number of basic schema structures are provided in the Minimal Schema and Basic Functional Schema, described in Appendix B and in

greater detail in [GOLD 88]. These schema "starter sets" are intended to support the functionality of the IRDS and help organizations begin to use the IRDS. While IRDS users will want to continue to use aspects of the Minimal and Basic Functional Schemas in many advanced IRD applications, the schema provided with the IRDS are designed to satisfy only a limited subset of users' needs. IRDS users should plan to expand and redefine their IRD schemas as appropriate for their unique applications.

Before initiating the IRD Schema Layer, various objects within the IRD Data Layer should be categorized into various entity-types, relationship-types and attribute-types. Table 5.2 and 5.3 illustrate the classification of entity-types and relationship-class-types, and relationship-types for the IRD data. There all the entities and relationships of the IRD Data Layer are grouped into entity-types, relationship-types and relationship-class-types according to the context of IRDS. New entity-types, relationship-types, and relationship-class-types are added when required to reflect the need of the study.

Table 5.2 Entity Groupings of IRD Data Layer

<u>IRD Entity</u>	<u>IRDS Entity-Type</u>
APPLICATION	SYSTEM
APPL_ACCOUNT	ACCOUNT
CPU	HARDWARE
DATABASE	DATABASE
DATABASE_ACCOUNT	ACCOUNT
DATABASE_BACKUP	OPERATION
DATABASE_FILE	PHYSICAL_FILE
DBMS	DBMS
EMPLOYEE	USER
INDEX	INDEX
NETWORK	NETWORK
OPERATING_SYSTEM	OPERATING_SYSTEM
PROGRAM	PROGRAM
SHUTDOWN_SCHEDULE	OPERATION
SOFTWARE_TOOL	SOFTWARE
TABLE	TABLE
TABLESPACE	LOGICAL_FILE

The organization of metadata about software objects within the prototype IRD Data Layer follows the Basic Functional Schema of the ANSI IRDS. In the prototype organization, the following entity types from the Basic Functional Schema are used: SYSTEM, PROGRAM, USER, ELEMENT.

Additional entity types are required to fully specify the prototype entities, these include:

ACCOUNT	Describes database account information
DBMS	Describes database management system
DATABASE	Describes occurrence of a database
HARDWARE	Describes computer hardware equipment
INDEX	Describe indexes for relation tables
PHYSICAL_FILE	Describes the physical file for database
LOGICAL_FILE	Describes the logical file for database
OPERATION	Describe database-related operations
OPERATING_SYSTEM	Describes the operation system use by the hardware
NETWORK	Describes communication network
SOFTWARE	Describes available software tools
TABLE	Describes relation tables

3

**PM-1 3½" x 4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT**

PRECISIONSM RESOLUTION TARGETS

PIONEERS IN METHYLENE BLUE TECHNOLOGY SINCE 1974

MicroD
CONVERSION

10000 COUNTY ROAD 8 BURNINGWELL, BURN BRIST, USA
TEL: 011 430 780 FAX: 012 430 7807 TLR: 0120000400

In the prototype, the Relationship Class Types from the ANSI IRDS's Basic Functional Schema are used: CONTAINS, RESPONSIBLE-FOR.

New relationship class types and relationship types are also required in the prototype. These new ones are used to fully capture the technical architecture under which a database operates:

ASSIGNS-TO	One entity is assigned to the other
ASSOCIATES-WITH	One entity is associated to the other
LINKS	One entity is linked to the other
OWNS	One entity owns the other
RESIDES-IN	One entity is resided in the other
USES	One entity is used by the other

Examples:

SYSTEM-USES-DBMS

NETWORK-LINKS-HARDWARE

DATABASE-CONTAINS-SOFTWARE

In the prototype, the relationships in the IRD Data Layer are represented unidirectional, not reciprocal. This is done to simplify the prototype. The relationships of the prototype IRD Data Layer are shown in Figure 5.3. Only a few relationships are fully populated in the prototype IRD Data Layer, because this is sufficient for the purposes of demonstrating the prototype.

Figure 5.3 Relationships between Entities for the Prototype's IRD Data Layer

Legend:

AT	Assigns-To	O	Owns
AW	Associates -With	RF	Responsible- For
C	Contains	RI	Resides -In
L	Links	U	Uses

Table 5.3 Relationships between Entities for IRD Data Layer

<u>Entity 1</u>	<u>Relationship</u>	<u>Entity 2</u>
APPLICATION	owns 1 or many owns 0 or many uses 1	APPL_ACCOUNT PROGRAM DBMS
APPL_ACCOUNT	owns 1 1 or many owns 1 or many	DATABASE_ACCOUNT TABLE INDEX
CPU	uses 1	OPERATING_SYSTEM
DATABASE	contains 1 or many contains 1 or many resided in 1 uses 1 contains 1 or many contains 1 or many contains 1 or many	DATABASE_ACCOUNT DATABASE_FILE CPU DBMS DB_BACKUP_INFO SHUTDOWN_SCHEDULE SOFTWARE_TOOL
DATABASE_ACCOUNT	assigns-to zero or many	TABLESPACE
EMPLOYEE	responsible-for 1 or many	APPLICATION
INDEX	resides-in one or many	TABLESPACE
NETWORK	links one or many	CPU
TABLE	resides-in one or many	TABLESPACE
TABLESPACE	associates-with 1 or many	DATABASE_FILE

In the prototype, the Attribute Types from the ANSI IRDS's Basic Functional Schema are used: DESCRIPTIVE-NAME, DESCRIPTION, DATE-ADDED, LAST-MODIFIED-BY, LAST-MODIFIED-DATE, LOCATION, etc. Additional Attribute Types are also required by the prototype to more fully specify the new entity-types or relationship-types. Some of these have been incorporated into the DBA Toolset's prototype IRD:

ACCESS-METHOD	Describes method to access database
STATUS	Describes the current status

IRDS Framework Required for DBA Toolset Prototype

The IRDS framework required for the initial DBA Toolset includes the basic elements to maintain entities and relationships, and the elements needed to support extensibility. Thus, the DBA Toolset Schema must contain at least the following entities [CSR 88]:

- Entity Type,
- Relationship Type,
- Attribute Type,
- Relationship Class Type,

and the following relationships:

- Entity__Type contains Attribute__Type,
- Relationship__Class_Type links Entity__Types, and
- Relationship__Type contains Attribute__Type.

The following elements of the IRDS standard are not critical to a first implementation of the DBA Toolset Prototype:

- Attribute Group Types,
- life cycle management,
- views, and
- security.

By relaxing the requirements for attribute groups the necessary framework can be easily mapped into a set of tables in the relational

model. It is not required that each component of the schema be stored in a separate table, they might be combined and retrieved via views, as illustrated in both the Dolk [DOLK 87] and the National Bureau Standard (NBS) [GOLD 88a] prototypes. The requirement that attributes can be dynamically added requires a DBMS that allow dynamic extension of its tables.

The meta-entity descriptions for the IRD Schema Layer are discussed in Appendix D. The descriptions for the IRD Schema Layer are also presented in Appendix D. The E-R model for the IRD Schema Layer is also illustrated in Figure D-2 of this Appendix. A minimal set of tables, with examples, to support this IRDS framework is given in Figure 5.4. The schema definitions for the IRD Schema Layer and the detailed schema design for the IRD Schema Layer are presented in Appendix D as well.

The IRD Schema Layer and the IRD Data Layer could be linked by a column (REF_TBL_NAME) in the IRD Schema Layer tables. This column serves as a pointer to the IRD Data Layer tables. The values of REF_TBL_NAME are the names of the IRD Data Layer tables in which the occurrences of the corresponding entities-type and relationship-types are stored. However, this feature is not currently implemented in this prototype.

Figure 5.4 Sample Relational Tables required for an initial Toolset Prototype

5.2.2 The MDAS Interface

The Interface to the DBA Toolset prototype consists of the menus by which the DBA executes all monitoring and administrative functions for the organization's databases. Through this MDAS Interface, the Catalog Extraction Programs are established and invoked, the Command Programs are executed and the Access Programs are run. In addition, the MDAS Interface provides the facility to administer the contents of the IRD. Finally it also allows the IRD administrator to administer the prototype toolset.

An important design characteristic of the MDAS Interface is that it be the sole method of accessing the IRD. This requires that a security module be an integral part of the interface and that the IRD be protected from any other method of access.

The construction of the prototype MDAS Interface uses the Oracle product SQL*Menu to fulfill the design requirements. SQL*Menu was used to implement a full menu-driven interface and also to address the controlled access/security requirements. The menu component consists of a list of choices, or menu items that the user can select to invoke a particular monitoring and administrative action. Menu items can cause commands to be executed, programs to be run or they may call screens or other lists of menu items.

The prototype MDAS Interface is a set of interconnected menus (a menu tree) from which the user may execute needed monitoring and administrative functions. A menu tree has a root menu, or main menu, connected to other menus, called submenus. A submenu is a menu that is called by a menu item in a higher level menu (the parent menu). Any submenu can also be a parent menu if any of its menu items call another submenu. This capability affords the designer the opportunity to group detailed lists of related functions into a series of submenus [ORAC 89]. A sample design of the main menu and submenus of the prototype MDAS Interface is presented in Figure 5.5. The detailed menu description and the menu program (MDAS.MNU) may be found in Appendix E1.

To control access and ensure security of the DBA Toolset, the MDAS Interface prototype contains provision to permit access based on the two types of roles:

- **MDAS_DBA** may access and execute all monitoring and database administrative functions, and query and report on IRD information;
- **MDAS_Administrator** may maintain the IRD and add to the functionality of the DBA Toolset. Other user roles can easily be added to the prototype if required. More details about the MDAS user roles are discussed in Section 5.2.7.

The implementation of an interface designed in such a manner results in a number of benefits:

- easy to learn, easy to use
- complexities are hidden
- minimize typing, syntactical errors
- improve security
- control access to those authorized
- reduce training, support costs.

Figure 5.5 Sample Main Menu and Submenus

Q/R DB Info	Cap DB Data	Oper on DBs	IPD Operation	NDRS Admin
FILE				
Query Report Database Information				
Application: NDRS		OSC>DBG>		Rep>

Q/R DB Info	Cap DB Data	Oper on DBs	IPD Operation	NDRS Admin
FILE				
<div> <div>DBS File Info</div> <div> TS Info. Table Info. Index Info. Acct Priv Info. Acct Quota Info </div> </div>				
Database File Info				
Application: NDRS		OSC>DBG>		Rep>

5.2.3 The Catalog Extraction Programs

The purpose of the Catalog Extraction Programs (CEP) is to provide the means by which the system catalogs of the organization's databases are accessed, relevant data is extracted, and that data is transferred to the DBA Toolset Information Resource Dictionary.

The design of the CEP's for the DBA Toolset prototype requires that only a single, centralized copy of each type of Extraction Program exist which can then be executed against any number of target databases. For example, if a CEP was established to be executed against 4 ORACLE (Ver.5) target databases, only 1 copy of the CEP need be existence (stored on the same node as the IRD). The CEP is invoked and executed through the creation of either a LOCAL or REMOTE batch process. By keeping only 1 copy of each type of CEP, maintenance and control of the CEP's is made considerably more effective. The prototype program (CR_CAT_EXTRACTION.SQL) to create the batch process is shown in Appendix E2.

In constructing the initial prototype CEP's, it was necessary to create a control table and install it in each target database. This table provides a row for each application within the database where data is to be extracted and transferred to the IRD. This control table is used to limit the amount of data being extracted and transferred. The schema

description of this table and the actual table schema are presented in Appendix E2.

To illustrate the use and potential of the Catalog Extraction Programs, 6 types of catalog extractions were considered within the prototype:

- database file data
- database user data
- fragmented table data
- user quota data
- index data of fragmented table
- partition/tablespace data

For each type of catalog extraction, a SQL program (script) was constructed which accesses the system catalog of the target database and writes the desired information to a temporary (or work) table. Since the DBA Toolset prototype accesses two different types of databases (Oracle Ver.5 and Oracle Ver.6), it was necessary to construct 2 SQL scripts for each of the 6 catalog extractions. The name and description of each SQL script and the actual SQL listing are presented in Appendix E2.

Each of the Catalog Extraction Program accesses certain tables within the system catalog of the target database in order to fulfill its

particular extraction criteria. For the six CEP's in this prototype, the list of system catalog tables accessed are given in Appendix E2.

The design and construction of the Catalog Extraction Programs is based on two important premises. A vendor's implementation of the relational model requires that there be a system catalog of metadata which describes and controls the contents of the database. The second premise is that the Structured Query Language (SQL) provides access both to the database and the system catalog. Therefore, the Catalog Extraction Programs may all be written in SQL allowing access to any vendor's relational database. Each CEP however, must account for differences in the system catalogs as implemented by the database vendors.

Once the data in the system catalogs have been accessed, extracted and written to a temporary table, the method of transferring that data was addressed. For the purpose of the prototype, the ORACLE utility Export/Import was used. This process consisted of 2 steps:

- Create Export file from data in the temporary tables;
- Import the file into the IRD.

As an alternative means of transferring the data, a second process was explored. This consisted of creating an ASCII file of the data and then using the ORACLE utility SQL*Loader to populate the IRD. This

latter process would have wider use since it would provide the means to operate on non-ORACLE databases residing on different types of hardware platforms.

The functions performed by CEP are illustrated in Figure 5.6.

The initial prototype of the DBA Toolset has successfully implemented the processes described above using the Export/Import mechanism and the results of the six types of data extracted are shown in a series of screen printouts in Appendix E2.

The screen interface (program CEP_EXTRACT.INP) for the catalog extraction process allows the authorized users to enter the required information. This screen layout and the screen program are in Appendix E2. The screen loads and access a temporary table called TEMP_CEP_PARM which is shown in Appendix E2. The detailed schema design for this table is also presented in Appendix E2.

To implement the prototype extraction process executing against a remote target database, it was necessary to construct a few additional routines which would facilitate this. These are summarized and presented in Appendix E2. The menus, screens, and program listings and sample outputs associated with the catalog extraction process are also presented in Appendix E2.

Figure 5.6 Catalog Extraction Program - Overview

5.2.4 The Access Programs

The Access Programs component of the DBA Toolset prototype provides the methods by which the DBA can access and analyze the consolidated information about the organization's databases. These are the DBA Tools so necessary to the DBA in order that he or she be able to access the status, availability, performance and integrity of all databases in use within the organization.

The Access Programs in this prototype allow the DBA to perform the monitoring and analysis functions on three databases residing in two nodes (as discussed in Section 5.0). The Access Programs consists of six screen programs and six reports which allow the DBA to monitor and analyze the metadata captured from the system catalogs of the three databases. The screen programs allow the DBA to query on the captured information about these three databases. The reports provide complete information on the captured data.

For example, to monitor the space availability in these three databases, the DBA query on the TABLESPACE screen (Figure 5.7). The DBA can enter query criteria on any enterable fields and then execute the query and the requested data will be displayed on the screen. The DBA can also perform a query on all records without entering any query criteria. To get a hard copy of the tablespace information, the

DBA can invoke the reporting option in the same screen and a report will be generated. A list of Access Program is presented in Appendix E3.

The Access Programs were constructed using two development tools from Oracle Corporation - SQL*PLUS and SQL*FORMS. Each Access Program is invoked by selecting the appropriate menu item from within the MDAS Interface. This design approach lends itself to extending the number and variety of functions which can be performed since to add a new function requires only the addition of a new menu item to the MDAS Interface which is then linked to the new function. The menu, screens, program listings and sample outputs associated with the Access Programs are presented in Appendix E3.

Figure 5.7 **TABLESPACE** Screen

MDBS		TABLESPACE (PARTITION) Information				1991-JUL-18	
DB ID	Database Desc.	Tablespace (Partition)	Free Bytes	% Free	# of Extents	Status	Date Extracted
7	5 CC1	DEER6	104752	99.0	1		18-JUL-1991
8	5 CC1	DEER4	104752	99.0	14		18-JUL-1991
9	5 CC1	DEER26	104752	99.0	1		18-JUL-1991
10	5 CC1	DEER24	104752	99.0	1		18-JUL-1991
11	5 CC1	DEER22	104752	99.0	1		18-JUL-1991
12	5 CC1	DEER20	104752	99.0	1		18-JUL-1991
13	5 CC1	DEER2	104752	99.0	1		18-JUL-1991
14	5 CC1	DEER12	104752	99.0	1		18-JUL-1991
15	5 CC1	DEER31	104752	99.0	1		18-JUL-1991
16	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
17	5 CC1	DEER30	104752	99.0	1	ONLINE	18-JUL-1991
18	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
19	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
20	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
21	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
22	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
23	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
24	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
25	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
26	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
27	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
28	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
29	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
30	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
31	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
32	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
33	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
34	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
35	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
36	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
37	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
38	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
39	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
40	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
41	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
42	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
43	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
44	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
45	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
46	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
47	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
48	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
49	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
50	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
51	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
52	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
53	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
54	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
55	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
56	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
57	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
58	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
59	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
60	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
61	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
62	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
63	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
64	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
65	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
66	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
67	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
68	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
69	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
70	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
71	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
72	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
73	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
74	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
75	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
76	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
77	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
78	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
79	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
80	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
81	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
82	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
83	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
84	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
85	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
86	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
87	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
88	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
89	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
90	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
91	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
92	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
93	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
94	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
95	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
96	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
97	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
98	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
99	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991
100	5 CC1	DEER3	104752	99.0	1	ONLINE	18-JUL-1991

Do you want to generate Tablespace Information Report (Y/N)?

Count: 25

(Replace)

5.2.5 The Command Programs

The Command Programs (CP) provide the means whereby the DBA may operate on remote target databases in order to perform the following administrative operations:

- enroll user;
- revoke user;
- alter user;
- grant access privileges;
- create public views;
- create public synonyms;
- perform export/import.
- enable/disable auditing

The design of the Command Programs requires that the DBA performs these operations from within the DBA Toolset and that it not be necessary for the DBA to physically log on to the remote target database. In other words, the DBA Toolset prototype provides a centralized center of monitoring and administration from which the DBA may execute a complete array of monitoring and administrative functions on any database in use within the organization.

To illustrate the use and potential of the Command Programs, this prototype considered the automation of one of the most frequent tasks a

DBA must perform -- that of enrolling a new user on a database.

Normally, the execution of this task would encompass the following steps:

- log on to remote node;
- log on to target database;
- key GRANT SQL statement or execute a predefined script to create the database account;
- log off target database;
- log off remote node.

Using the DBA Toolset prototype and the associated Command

Program, the process becomes much simpler and more efficient:

- DBA selects "Operate on Databases"
- DBA selects "Enroll Users" function;
- DBA supplies User Id, Node_ID, Database_ID and password;
- DBA sends Command Program.

The functions performed by this Command Program are illustrated in

Figure 5.8

Figure 5.8 Enroll User - Command Program

The screen interface (ACCT_REG.INP) for this prototype Command Program allows the DBA to enter the required parameter information. This screen is presented in Appendix E4. The screen loads and accesses a temporary table called TEMP_ACCT_REG which is shown in Appendix E4. The detail schema design for this table is also presented in Appendix E4.

To support the prototype Command Program to execute against a remote target database, it was necessary to construct two additional routines (CR_USER_COM.SQL, COPY_SUBMIT.COM) would facilitate this. These are also summarized in Appendix E4. The menus, screens, program listings, and sample outputs associated with the Command Programs are presented in Appendix E4.

5.2.6 The IRD Operation Programs

The IRD Operation Programs provide the administrator of the IRD with the ability to query, report, modify and/or extend the data stored in the IRD. The IRD Operation Programs also allow other users such as DBAs, project leaders, etc., to query and report on the IRD data.

There are two types of screens in the prototype IRD Operations: the query/report screens and the maintenance screens. The MDAS Administrator and other users use these programs to query or report on the Schema and Data IRD Layers. The maintenance screens also allow the MDAS Administrator to register new databases, new nodes, new systems, new programs, etc. as well as add or extend new entity types, attribute types, relationship class types, relationship types, etc. Thus one or two screens were constructed on each of the major IRD Schema and Data Layer's table.

For example Figure 5.9 shows the various options available under the prototype's IRD Operation option. Only the IRD (MDAS) Administrator will see all four items available to him/her under this option. All other users only see two query/report items: Q/R IRD Schema Layer and Q/R IRD Data Layer.

The IRD Operation Programs were constructed using 2 development tools from ORACLE Corporation - SQL*Plus and SQL*Forms. Each IRD Function program is invoked by selecting the appropriate menu item from within the MDAS Interface.

The query/report screens are designed to allow the user to query on all or a specified data from the IRD layers. Query criteria can be entered through the querable fields before executing a query. A reporting option is also available on the screen if the user wants to generate a detailed report.

The maintenance screens are designed to allow insertion, deletion and modification of IRD data. In the prototype, the data integrity checks are performed on insertions to validate the entry data. These validations are done using Oracle's SQL*Forms triggers (a set of user-written processing commands) to check against the master tables (e.g. Entity Type, Database tables).

The IRD Operation Programs are briefly described in Appendix E5 and the menus, screens, programs, and sample outputs related to the IRD Operations can be found in Appendix E5.

5.2.7 MDAS Administration Programs

This component provides the "housekeeping" facilities to the administrator of the DBA Toolset. This menu item is only available to the IRD Administrator. That is this item will not appear as a menu item when other types of user sign on to the DBA Toolset. This component permits the IRD administrator to enroll, revoke, and alter user access privileges to the DBA Toolset. There are two user roles in the DBA Toolset prototype: MDAS_ADMINISTRATOR, MDAS_DBA.

The MDAS_ADMINISTRATOR is the IRD administrator who is responsible for the above mentioned tasks. The administrator is also responsible for maintaining the metadata in the IRD Schema Layer using the IRD Function programs described in previous section.

The MDAS_DBA is the DBA who can perform catalog extraction processes; load the captured system catalog data into the IRD Data Layer and query and report on the data stored in the IRD. The DBA can also perform operations on other databases through the DBA Toolset.

To facilitate access control over the DBA Toolset, two tables containing the toolset user and role information are required. The description of these two table and the detailed schema can be found in Appendix E6.

The MDAS Administration Program were constructed using two development tools from Oracle Corporate - SQL*Plus and SQL*Form. Each Administration Program is invoked by selecting the appropriate item from within the DBA Toolset. The programs related to the application administrations are given in Appendix E6.

The design of the prototype allows menu user types (such as Query only) to be easily added to the DBA Toolset by adding the new roles to the MDAS Interface. The menus, screens, and programs associated with the application administration can be found in Appendix E6.

5.3 Populating the DBA Toolset IRD

Data about an organization's database information provides a baseline for developing effective Database Administration. However, obtaining accurate data can be a major stumbling block to IRDS implementation because the collection effort is often perceived as relying heavily on manual data entry, a time-consuming and labour-intensive task [KUIL 90]. For the prototype, the tables which contain the captured database information are the largest tables in this prototype. These tables are populated by the catalog extraction process which load the captured data into the prototype's IRD Data Layer.

Since this study only implements a small subset of the ANSI IRDS, the amount of data related to the Prototype is limited. Hence all the IRD tables are populated by entering the metadata manually through the maintenance screens of the IRD Operation Programs. For example, to add a new entity-type "DATABASE", we need to perform the following:

- Add "DATABASE" entity type to the table ENT_TYPE through the maintenance screen ENT_TYPE.INP
- Add attribute type information relating to "DATABASE", to the table ENT_TYP_CONTAINS_ATT_TYP through the screen ENT_TYP_CONT_ATT_TYP.INP.

e.g. "DATABASE" contains attribute "DESCRIPTION"

"DATABASE" contains attribute "NAME"

"DATABASE" contains attribute "LOCATION"

"DATABASE" contains attribute "ACCESS-METHOD"

etc.

- Add Entity and Relationship related to "Database" to the

ENT_REL table through the maintenance screen

ENT_REL.INP.

e.g. DATABASE-contains-SOFTWARE

DATABASE-contains-OPERATION

CHAPTER 6

THESIS SUMMARY

6.1 Contributions

This thesis presents several contributions to the field of Database Administration. It has carried out a detailed study of different aspects of database administrative functions and grouped them into several categories. This study also identified those database administration functions that are feasible for automation. Also studied were the capabilities of a representative set of DBA tools in the market for managing several relational database management systems. These existing tools only handle one type of DBMS and operate on a single database at a time. It has been pointed out that current database administration tools are unable to deal with multiple database environments and are more suitable for organizations with a small number of databases. This study emphasized the importance of managing multiple databases with a single DBA Tool.

A new type of DBA tool has been proposed by this study: a Multiple Database Administration System (MDAS) which provides a toolset that permits the DBA to manage all databases on the enterprise network regardless of DBMS type or the type of hardware upon which they operate. The proposed DBA Toolset is composed of a centralized repository which contains critical information about the enterprise's

databases and their related metadata; and a set of tools which allow the DBAs to monitor and operate the enterprise's databases.

The proposed DBA Toolset's centralized repository employs the ANSI IRDS framework. Under this approach, the metadata could easily be extended to capture additional information which may be required by the database administration function. Using an IRD, the DBA can describe, control, document, maintain, manage and track all critical information on all databases (including those from different DBMS vendors) residing on different hardware platforms of an enterprise. The IRD also allows the DBA Toolset to self-document information such as DATABASE-USES-DBMS (which database uses which DBMS), DATABASE-CONTAINS-SOFTWARE (which database uses which software), etc.

The thesis shows how new metadata can easily be added to the IRD. For instance we have discussed the addition of new entity types such as: OBJECT_BASE, KNOWLEDGE_BASE (in addition to DATABASE). The extensibility of the IRD allows the DBAs to meet the needs of the changing world and operating environments.

It has been shown that the multiple database administration capabilities of the proposed DBA Toolset can readily overcome some of the major shortcomings of current DBA tools such as reviewing database

status one at a time; working with only one type of DBMS; allowing the DBA to perform operations on one database at a time.

An important result of this work has been the description of mechanisms to bridge heterogeneous relational databases that are running on various hardware environments. By extracting various types of information from various types of databases running on different hardware and loading them into the IRD, the DBA only needs one set of tools to review the status of all databases within the enterprise. The IRD can also contain the information necessary to describe and control the execution of the catalog extraction processes.

Another important result of this study has been the successful demonstration of sending database commands to different databases residing at other nodes without the need to manually sign on the target database and node. Again the IRD information was used to identify the access information for the target databases and sign on automatically through a batch process. This approach saves the DBA considerable time especially in a large organization with many databases.

Finally, the ideas developed in this thesis were examined by development of a prototype for a multiple database administration environment in the ORACLE and VAX/VMS platform for a large organization (AECL Research Company). Early feedback from the

organization is encouraging and the DBA is able to monitor the status of a number multiple databases using the DBA tool. Furthermore, the DBA also benefits by using the toolset to send database commands (eg. user enrollment) to different production databases. Moreover, all the captured system catalog information can be made available to management personnel, data administrators, project leaders, etc. without the need to give out DBA privileges to those individuals.

6.2 Limitations and Constraints

Despite the potential capabilities to manage multiple databases, the MDAS tool currently has certain limitations and constraints of which both the would-be implementor and the user must be aware.

The MDAS concept depends entirely on a centralized IRD and a failure with the central point could stop all operations to and from the IRD. However, normal operations on the databases could be continued by manually signing on to the target database and node.

The current MDAS also uses a passive IRD in which the data are brought in from other system catalogs in a regular fashion (such as downloading the information periodically using a batch operation). Therefore, some of the metadata captured in the IRD may not be quite current. However, currency is not a crucial DBA requirement for most of the information such as trend-monitoring, space usage, free space availability, etc., and this constraint is therefore not serious. An alternative is to extract the metadata more frequently as discussed in Section 4.3.4, or on demand as required.

Another factor is that if the IRD is used to contain all possible information in the system catalogs and if the catalog extraction processes were performed in a more frequent manner, the volume of information may be far too large to be transmitted over the network. This needs to

be balanced against the frequency of change of the catalog information at the databases themselves, and the network traffic requirements should also be considered. In practice, most databases on workstations within a network will probably exhibit low rates of change in their database catalogs and will therefore require only infrequent monitoring of their catalogs. The most appropriate mechanism would be one which automatically adjusts monitoring frequency based on the rate of change of database catalogs being monitored.

MDAS requires a separate set of catalog extraction programs and command programs for each type of database (e.g. ORACLE Ver.5, ORACLE Ver.6). The number of catalog extraction programs and command programs would increase if there are many different types of DBMS (e.g. INGRES, Rdb, INFORMIX, SYBASE, etc.) running on various hardware equipment within the network. Modifications to these programs may be complex and labour-intensive. However some of the complexities may be eased with the self-documentation features of the IRDS.

The prototype requires a control table (or filter) residing at the target database to limit the amount of information to be extracted. If the DBA decides to modify the control parameters, he must sign on manually to the individual database. This method should be reviewed

and may be redesigned in such a way that the control table is dynamically created at the target database within the batch process of the batch job file.

Currently, the MDAS prototype captures the same pieces of information from all target databases. This does not provide the degree of selectivity for the DBA to meet his or her own needs. Again, the catalog extraction process could be re-designed to accept screen parameters from the DBA before generating the batch job file for the target database.

Obtaining accurate data for the initial population of certain tables within the IRD can be a major impediment to successful IRDS implementation because the data capture effort often relies heavily on manual data entry, a time-consuming and labour-intensive task. This prototype used both automated and manual approaches to populating the IRD. However, for a full implementation of MDAS, the manual approach is not appropriate. Extraction software is required to automate a significant portion of this task.

Finally, the above factors suggest that an MDAS implementation utilizing object-oriented technology would be a fruitful area for further investigation. Such an approach would model a database as an object, with a set of methods to support the required MDAS behaviour identified

in the current MDAS prototype. Such an implementation would consist of the same IRD-backed centralized system, but would model the catalog extraction programs and the command programs that control the database as methods associated with objects of type database. MDAS would generate the method implementation based on database type and hardware type, and place the method implementation at each database for subsequent invocation by batch processes or remote procedure call. Requirements for variability in behaviour such as extraction frequency etc. discussed above could be handled by having MDAS generate the appropriate control parameters to govern behaviour during the code generation process that generates the method implementation placed at each database instance for subsequent use by MDAS.

6.3 Conclusions

Based on the work in this thesis, the following observations and conclusions are made:

- 1) The MDAS approach shows excellent promise of being able to use a single DBA tool to manage multiple databases within an organization. If the identified limitations and constraints discussed in previous section are resolved, the MDAS concept will be a powerful DBA tool. The most important design objective of MDAS is to provide a single tool which can monitor and analyze the status of all databases in the enterprise regardless of database type or the type of hardware upon which they operate. Even though the prototype only dealt with two different versions of the same RDBMS vendor (ORACLE) operating on the same hardware platform, it did demonstrate that the objective is achievable and the proposed concept should be investigated further to deal with real heterogeneous databases operating on different hardware platforms.
- 2) Another important design objective of the MDAS approach is to provide extensibility of the data dictionary to allow for changing database administration requirements. The proposed DBA Toolset employs the ANSI IRDS approach for its centralized data repository. This approach can be easily extended to capture any type of data and processes that

relate to database administration. It also provides users with access to a logically centralized repository of data about all relevant information resources within the organization. The data within the IRD also describes the composition of, and relationship between, information resources associated with database administration.

3) The IRDS provides a comprehensive specification for a data dictionary.

It will, at minimum, provide a framework for the discussion and evaluation of data dictionary products. Very likely, many vendors will provide software which conforms to the standard. A number of vendors have already stated informally that they intend to do this [WERT 89].

Some of the benefits of using an IRDS include: improve the identification of existing resources; aid development, modification and maintenance of systems; allow self-documentation; etc.

4) Extensions to the ANSI starter set are required to fully describe the information related to the database administration environment. These extensions are discussed in Section 5.2.1.2.

5) As CASE technology evolves and supports IRDS and SQL2 standards, the efforts to build the catalog extraction programs and command programs for the organization will be reduced.

6.4 Future Development

The Multiple Database Administration System concept appears to be an extremely rich area of follow-on research as evidenced by the identification in previous Sections of numerous areas which would benefit from further investigation. Their complexities preclude further analysis as part of this particular thesis. However, their resolution is a necessary prerequisite to the construction of a full function Multiple Database Administration System. We have briefly summarized the most significant ones below:

- 1) Enhance the prototype to operate on heterogeneous databases running on different hardware platforms;
- 2) Determine the full implementation of MDAS at AECL Research Company;
- 3) Investigate the possibility of generic catalog extraction programs which could operate on any type of RDBMS;
- 4) Investigate the possibility of generic command programs which could generate commands for any type of RDBMS;
- 5) Provide selectivity (or filters) for the catalog extraction process to limit the amount of data to be extracted and what type of data should be extracted;

- 6) Determine the strategies for moving the prototype IRD to a work station environment (e.g. ORACLE under OS/2);
- 7) Investigate the administration of network, hierarchy databases using MDAS;
- 8) Design and development of MDAS for a distributed database environment;
- 9) Investigate the application of MDAS to future generation of databases such as Object-Oriented, Multi-Media, etc.;
- 10) Investigate an implementation of MDAS using object-oriented technology.

REFERENCES

.

[ANSI 88]

"American National Standard for Information Systems - Information Resource Dictionary System (IRDS)", ANSI Inc., X3.138-1988.

[BARK 89]

R. Barker, "CASE*METHOD Entity Relationship Modelling", Addison Wesley, MA, 1989.

[BEST 84]

H. Bestougeff, "Designing the Database Management System", Elsevier Science Publisher B.V. (North-Holland), 1984.

[CERI 84]

S.Ceri, G. Pelagatti, "Distributed Databases Principles and Systems", Computer Science Series CERI/PELAGATTI, McGraw-Hill Inc., 1984.

[CHEN 88]

C.W. Chen, "The DBA's Changing Role", Database Programming & Design, October 1988.

[CSR 88a]

Computer Science Research Department, Lawrence Berkeley Laboratory, "Data Encyclopedia Architecture for Army Information Management", Report LBL-25137, June, 1988.

[CSR 88b]

Computer Science Research Department, Lawrence Berkeley Laboratory, "Specifications for Version 1.0 of the Army Data Encyclopedia Architecture for Army Information Management", Report LBL-25746, September, 1988.

[CSR 88c]

Computer Science Research Department, Lawrence Berkeley Laboratory, "Usage Notes - Army Data Encyclopedia Textbook Version 0.5", Report LBL-26241, November, 1988.

[CSR 88d]

Computer Science Research Department, Lawrence Berkeley Laboratory, "Executive Summary - Data Encyclopedia Architecture for Army Information Management", Report LBL-25476, June, 1988.

[DATE 81]

C.J. Date, "An Introduction to Database Systems", Third edition, Addison Wesley, MA, 1981.

[DOLK 87]

D. Dolk, R. Kirsch II, "Relational Information Resource Dictionary System", Comm ACM 30, 1 (January 1989), pp. 8-61.

[EASY 89]

"easyDBA User's Guide", Version 1.0, Dimeric Development Corporation, U.S.A. 1990.

[GEY 88]

F. Gey, "A Data Encyclopedia Architecture within An Extended IRDS Framework", Proceedings of the FIPS Workshop on IRDS Application, pp.63-85, December, 1988.

[GOLD 87]

A. Goldfine, P. Konig, "A Technical Overview of the Information Resource Dictionary System", (Revision 1), Center for Programming Science and Technology, National Bureau of Standards, April 1987.

[GOLD 88a]

A. Goldfine, "Proceeding of the Federal Information Processing Standards (FIPS) Workshop on Information Resource Dictionary System (IRDS) Applications", National Institute of Standard and Technology, December 1988.

[GOLD 88b]

A. Goldfine, "Dictionary System (IRDS) Application", Proceedings of the Federal Information Proceeding Standards (FIPS) Workshop on Information Resource, December 1988.

[GOLD 88c]

A. Goldfine, "Using the Information Resource Dictionary Command Language", Second Edition, National Bureau of Standards, January 1988.

[GOLD 88d]

A. Goldfine, P. Konig, "Technical Overview of the Information Resource Dictionary System (Second Edition), National Bureau of Standards, NBSIR 88-3700, U.S. Department of Commerce, Gaithersburg, MD, January 1988.

[HABE 88]

B. Haberkamp, "Office of the Army Chief of Staff", Proceedings of the Federal Information Processing Standards (FIPS) Workshop on Information Resource, December 1988.

[INMO 81]

W.H. Inmon, "Effective DATA BASE Design", Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1981.

[ISO 89a]

"Information Resource Dictionary system (IRDS) Service Interface Working Draft", Revision 9, ISO, July 1989.

[ISO 89b]

"Information Processing Systems - Information Resource Dictionary System (IRDS) Framework", ISO, September 1989.

[JACK 88]

G.A. Jackson, "Relational DATABASE Design with Microcomputer Applications", Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1988.

[JAQU 87]

D. Jaqua, "DBA Companion An Oracle Security Management System", Oracle International User Grouping Conference, Orlando, Florida, September 1987.

[KUIL 90]

S. Kuiler, M. Laubacher, "Using Oracle and SQL*Forms to Prototype an Information Resource Dictionary System (IRDS)", Oracle International User Conferences, Anaheim, CA, September 1990, pp. 393-398.

[LAW 88a]

M.H. Law, "Guide to Information Resource Dictionary System Applications: General Concepts and Strategic Systems Planning", National Bureau of Standards Publications ,pp. 500-512, April 1987.

[LAW 88b]

M.H. Law, "Guide to IRDS Applications : General Concepts & Strategic System Planning", Proceedings of the Federal Information Proceeding Standards (FIPS) Workshop on Information Resource, December 1988.

[LEFF 89]

B. Leffew, "Database Sizing: Performance Implications and Automated Space Management Support", ORACLE Corporation, 1989.

[LOFT 89]

J.P. Loftis and W.E. Friggle, "Building a User-Friendly Data Dictionary System in Oracle", Oracle International User Groups conference, Dallas Texas, October 1989, pp. 439-444.

[LYON 76]

J.K. Lyon, "The Database Administrator", John Wiley & Sons, Inc., New York, 1976.

[MALA 88]

C. Malamud, "A Prototype Implementation of the Information Resource Dictionary System", Ingres User Association, 1988.

[MART 77]

J. Martin, "Computer Data-Base Organization", Second edition, Prentice-Hall Inc., Englewood Cliff, New Jersey, 1977.

[MCGO 90]

D. McGovern, T. Negriz, C. Reid, "Text and Graphics in Relational Databases", DB/EXPO'90, The National Database Exposition and Conferences, San Francisco, CA., March 1990.

[MURR 88]

M.C. Murrphy, "Integrity Constraints in the IRDS/ADE", December 1988.

[ORAC 88]

"ORACLE RDBMS Database Administrator's Guide", Version 6.0, Oracle Corporation, California, 1988.

[PRAB 90a]

M. Prabandham, D. Perschbacher and C. Wessel, "The Role of the IRDS", Database Programming & Design, April 1990, pp. 41-47.

[PRAB 90b]

M. Prabandham, W.J. Selfridge and D.D. Mann, "A View of the IRDS Reference Model", Database Programming & Design, March 1990, pp. 41-53.

[RUMB 84]

J. Rumble, V. Hampel, "Database Management in Science and Technology", Elsevier Science Publishers B.V. (North-Holland), New York, 1984.

[RUSS 84]

J. Russell, D.E. Cullen, "Implementing and Using the Database Management System", Elsevier Science Publishers B.V. (North-Holland), 1984.

[STON 90]

M. Stonebraker, "3-Dimensional DBMSs", DB/EXPO'90, The National Database Exposition and Conferences, San Francisco, CA., March 1990.

[WERM 89]

A. Werman, "Automating the Mundane", Database Programming & Design, December 1989, pp. 34-44.

[WERT 89]

C.J. Wertz, "The Data Dictionary Concepts and Uses", QED Information Sciences, Inc., Wellesley, Massachusetts, 1989.

[WINS 89a]

P. Winsberg, "The IRDS Standards: Interpretation and Forecast", Course Tutorial, 1989.

[WINS 89b]

P. Winsberg, "Dictionary Standards: ANSI, ISO and IBM", Course Tutorial, pp. C-175, 1989.

APPENDIX A -- Survey of DBA Tools

Contents

ORACLE DBA Tools

SQL*DBA, Oracle Corporation, California
DBA*Assist, Oracle Corporation, California
easyDBA, Dimeric Development Corporation, California
DBA Companion, SQL Solution Inc., Massachusetts
DBA*MASTER, Management Information Technology Inc.
Database Analyzer, SQL Solution Inc., Massachusetts
Resouce Manager, SQL Solution Inc., Massachusetts

INGRES DBA Tools

SYSBASE DBA Tools

SA Companion, SQL Solution Inc., Massachusetts

DB2 DBA Tools

DASD Analyzers
Application Mirgators
Security Managers

ORACLE DBA Tools

SQL*DBA, Oracle Corporation, California

ORACLE RDBMS provides a tool called SQL*DBA to help database administrators to manage and monitor a database. SQL*DBA can be used by DBAs to perform some of the major tasks such as: database creation; start and stop a database; monitor realtime use and performance of a database; perform backup and recovery of database; and execute any SQL statement. The DBA has to enter the proper SQL*DBA command in order to perform the required task. SQL*DBA can be run in interactive or batch mode. This tool also allows the DBA to connect to other databases (provided that SQL*NET is also installed) on nodes of the communication network. Thus the DBA can use SQL*DBA to perform some of the DBA tasks on a remote node.

DBA*Assist, Oracle Corporation, California

ORACLE also has another DBA tool, DBA*Assist, which can be purchased separately. This tool can be used to assist the DBAs in performing some of the DBA functions such as: database control (e.g. enables/disables system and table auditing); user maintenance (e.g. adds and changes users; grants and revokes user privileges; changes user space allotments); archiving (e.g. starts and stops archiving; lists archiving activities); tablespace control (e.g. sets online/offline status; maintains tablespaces); and disk space management (e.g. builds command files to create tables; prints disk space reports; analyzes utilization by tablespace, user, and tables; dynamically resizes and rebuilds tables). This tool automates some of the activities related to these areas, enabling the DBA to administer the database system without familiarity with SQL syntax. DBA*Assist is a form-based program which prompts the DBA for information and presents results.

easyDBA, Dimeric Development Corporation, California

easyDBA is an integrated set of database management tool to maintain ORACLE version 6 databases. It ties together with a menu-driven interface. Using these tool, the DBA has full control of the database objects and files. All of the requested objects are displayed on the screen without entering a single line of SQL. This tool can also be used by developers and end users to maintain their database objects. This tools is useful to DBA because it eliminates the tedium involved in performing the day-to-day administration tasks. Since this the tool is interactive DBA need not worry about whether the information displayed is current.

easyDBA automates many DBA functions such as: maintenance of database objects and database files; storage space management; and user maintenance. It provides tools to handle these DBA functions. These tools include table tools, view tools, synonym tools, index tools, grant tools, cluster tools, sequence tools, user maintenance tools, space utilization tools, tablespace tools, rollback segments tools, and database files tools [easy 89].

For instance, the free space and fragmentation tools provide the DBA a quick idea of the overall status of the database with regard to the amount of free space remaining as well as the degree of fragmentation.

For tables and indexes, the DBA can view how much space is being used individually or combined. Fragmentation statistics are also reported for tables, indexes, and rollback segments.

easyDBA also works with ORACLE's SQL*NET to allow the DBA to maintain databases on remote systems.

DBA Companion, SQL Solutions Inc., Massachusetts

This tool enables DBAs and developers to manage security, source code, users, applications and storage capacity for ORACLE environment.

The DBA Companion automates database table access privileges for users by scanning the application source code and automatically extracting table access requirements for each application objects such as screen, reports, program, etc. The DBA then defines roles for users and their application objects. When new database users are enrolled, the DBA simply assigns them to the appropriate role and all the access privileges or grants are automatically issued, audited, managed and reported on. The tool also has a dynamic run-time menu system which only displays the menu options for applications, screens, reports, etc. to which a user has access [JAQU 87].

This tool also provides an alternate database account facility to prevent users from performing insert, update and delete operations through SQL.

This tool also assists DBAs and developers in storage capacity management by providing trend-based analysis of database space usage, fragmentation and chaining, utilities for migrating from development to test to production, and a host of other productivity utilities.

This tool also provides where-used, impact-of-change and configuration control facilities. It comes with source code scanners which allows the developers and analyst to see the inter-relationships between all source code modules and files. A complete cross reference facility is also available for each database object. Thus allows developers and DBAs to see all modules that are affected by the change on a database object.

DBA*MASTER, Management Information Technology Inc, California

DBA*MASTER is a database administration's tool for the **ORACLE** database. It is designed to help the DBA with his/her most demanding responsibility, tuning the database system for optimal performance. It monitors the performance of the database, highlights potential problems, provides proposed solutions, and helps the DBA to manage resource overheads. This products consists of four modules: the Data Collector, the Analyzer, the Reporter and the Advisor. These modules automate different tasks.

The Data Collector automatically collects system activity data from both **ORACLE** and the operating system in predetermined intervals. The Analyzer pin-point performance problems such as: bottlenecks, locks, inefficient programming, **ORACLE** resources and computer resources overload. The Reporter provides reports of the system performance and of its components and suggests necessary tuning information of the **ORACLE** system. The Advisor provides advice pertaining the various aspects of system's performance such as database parameters, application tuning and utilization of the computer resources.

This product allows the DBA to detect problems in the applications, problems with the use of operating system and its resources and problems with **ORACLE** tuning. With the suggestions of the Advisor,

A-10

the DBA can perform his/her related duties while requiring less time and expertise.

Database Analyzer, SQL Solutions Inc., Massachusetts

The Database Analyzer is an in-depth analysis and exception reporting tool for ORACLE databases. It examines over 120 ORACLE database objects and attributes to produce exception reports and interactive screens that identify the portions of the ORACLE database environment requiring immediate attention.

It assists the DBA in accessing additional, detail information which the DBA may use to gain further insight about a specific problem area. Such information serves to detect and correct problems as they occur, and will aid the DBA in identifying potential problems well before they require corrective action.

When the analyzer detects certain problem areas, such as a number of widowed synonyms or grants still residing in the database, it will offer the DBA the option to automatically remove them from the database.

It also allows the DBA to monitor the critical operational control areas of security management, auditing, performance analysis and tuning, space management and capacity planning.

It also provides reports and interactive screens covering every aspect of the ORACLE database management.

Resource Manager, SQL Solutions Inc., Massachusetts

The Resource Manager is a storage management and capacity planning tool for the ORACLE RDBMS. It allows the DBAs to efficiently utilize storage space, identify and correct both fragmentation and chaining problems, and ensure optimal database performance. The DBA can also plan for future database expansion by monitoring the growth rates of rapidly evolving database environments.

There are four components within this product: the Recorder, the Reporter, the Expert and the Assistant:

The Recorder logs information about the objects in the database, giving the DBA immediate access to accurate space utilization information, both current and historical.

The Reporter generates a variety of reports based on the information stored by the Recorder, and plots this information in presentation format. The reports generated include statistics, totals, histograms and several trend analysis reports.

The Expert utilizes the information stored by the Recorder and locates anomalies in the storage architecture such as; which tables are candidates for striping, which indexes are redundant, which rollback segments are nearing MAXEXTENTS, etc.

The Assistant automates the process of resizing ORACLE objects. It eliminates the amount of time in calculating and applying accurate space definitions and storage clause to all objects. It also performs many tasks essential to maintaining an efficient database, such as moving objects from one tablespace to another, dropping and renaming objects, and eliminating fragmentation.

INGRES DBA Tools

INGRES RDBMS provides some utilities within the INGRES DBMS Server allowing the DBA to perform some major DBA functions. The Intelligent Optimizer allow DBAs to gather database statistics. This optimizer normalizes queries to a "canonical" form before starting optimization so that performance is not dependent on the particular phrase of a query. It understands two queries have identical meaning despite their different phrasing. It frees the DBAs from tuning application source codes due to its syntax-independent.

The Resource Control System allows the DBAs to set limits on return rows and disk I/O and it checks limits and aborts offending queries before they are even started. This gives a method to stop runaway queries before they happen.

The Access Control System allows the DBAs to perform group and application-level permissions, to limit access to certain data control language statements, and to grant resource control permissions to groups or applications.

Moreover, INGRES/STARView is a distributed database administration monitor which allows unified administration of distributed data. Thus allows centralized management of dispersed information resources. The DBA may directly log into particular distributed or local

**databases through the STARView monitor to administer, create, destroy
and modify their contents.**

SYBASE DBA Tools

SYBASE provides **Data Workbench**, which is an interactive, visually-oriented tool that includes an array of utilities designed to help the DBA with data administration, data dictionary access and bulk loading. The Data Dictionary displays detail information about databases and their objects and it also allows easy access via a user interface. The Data Workbench also helps the DBAs and analysts with database design, maintenance and tuning by providing tools to design and maintain databases, to monitor database activity, to check database consistency. It also provides a bulk data Transfer and conversion utility that copies and loads data on magnetic tape from disk files at very high speeds. While data is being transferred, users continue to have full use of the system.

SA Companion, SQL Solution Inc., Massachusetts

SA Companion provides a window-driven environment to automate the major functions of SYBASE system administrator. It helps the system administrators to manage complex SQL Server network easily without learning SYBASE-specific syntax.

It also organizes the system administration functions into five categories, each accessible from the primary menu: Servers, Devices, Users, Databases and Reports.

The Server Management menu allow users to connect, install, configure, examine and recreate entire servers, as well as manage user processes. The Device Management menu allow users to list, add, drop and mirror devices. The User Management menu lets users to list, add, delete users, groups and passwords. The Database Management menu let the users to list, create, delete, modify and recreate databases and associated objects. The Report Management allows the generation of parameter-driven reports on the status of various system objects.

DB2 DBA Tools

Four of the most important categories of DB2's DBA tools are storage and DASD management, application migrators, catalog management, and security management [WERM 89].

DASD Analyzers.

Some available products:

- RC/DB Analyzer, Platinum technology, Lombard, Ill.
- DASD Manager, BMC, Sugar Land, Texas
- DBC Inform/DB2, Schumann Consulting, Englewood Cliff, N.J.
- DBC Space/DB2, Schumann Consulting, Englewood Cliff, N.J.
- SMU2-DB2, CDB, Houston, Texas.
- Spaceman-DB2, CDB, Houston, Texas.

These products analyze DB2 data sets and attempts to identify problems that reduce the efficiency for database access or might prevent access to data. These products look at DB2 data set and review the DASD implications of their structure. They read wither the space-map pages of a data set or every page of the data set. Using the gathered information, the products advise on available space for updates, the amount of data forced to an overflow page, and distances between logically sequential pages. The products also gather information on the disk-drive volume table of contents, which can be use to determine

whether device placement is appropriate and whether the data set is in extents (and, if so, where they are).

Application Migrators.

Some available products:

- DB2 Migrator, CDB Software Inc., Houston, Texas.
- Migrator, Platinum Technology, Lombard, Ill.

An application migrator provides facilities to move an application from development to test and then to production in a controlled manner. This involves recreating databases, tablespaces, tables, views, synonyms, and VSAN clusters, creating any authority required, and copying data where appropriate. This process is usually complicated by differences in object sizes and in naming conventions between environments.

Catalog Managers.

Some available products:

- DB/Excel, Reltech Products
- Intellect/DB2, AI Corp.

Catalog managers allow DBAs to review DB2 catalog information.

They provide panels to generate data definition language and data control language and come with value-added features such as storage-space calculators for object sizing. In view of the limitations of the DB2 catalog for design purposes, these could be considered very low-end CASE tools.

These products are designed as front ends to DB2, allowing developers to check database layouts, DBAs to submit utilities, and neophytes to get a feel for DB2.

Security Managers.

Some available products:

- RC/Secure, Platinum Technology, Lombard Ill.
- DB/Secure, Systems Center Inc., Reston, Va.

Security managers were developed in response to DB2 v. 1's security control limitations. These tools help DBAs assemble collections of authority into logical groups and grant and revoke authority on a group wide basis. Security managers also help produce reports of authority, deal with cascading privilege revocation, and clean up duplicated authority.

APPENDIX B -- Overview of ANSI IRDS

Contents

OVERVIEW OF ANSI IRDS

- Background
- What is an IRDS?
- Purpose of the IRDS Standard
- IRDS Benefits
- The Architecture of the IRDS

FEATURES OF THE ANSI IRDS

- The ANSI IRDS Data Model
- The ANSI IRDS Architecture
 - IRD Schema Description Level
 - IRD Schema Level
 - IRD Data Level
 - Production Data Level
- The IRDS Contents
 - The IRDS Core
 - Minimal Schema
 - Basic Functional Schema
- ANSI IRDS Facilities
- ANSI IRDS Interfaces

OVERVIEW OF ANSI IRDS

Background

The American National Standard Institute (ANSI) and the National Institute of Standards (NIST, formerly National Bureau of Standards) have been hard at work since 1980 on a standard for dictionary software. The ANSI effort began with the Accredited Standard Committee X3H4 responsible for developing the standard for an Information Resource Dictionary System (IRDS). While the NIST focused on the development of a Federal Information Processing Standard (FIPS) for Data Dictionary Systems. The two efforts came together in 1983 when ANSI adopted FIPS for an IRDS and merged work with NIST.

Meanwhile, the International Organization for Standardization (ISO) has been considering a parallel IRDS standard. The ISO effort began in 1984 and was based on the ANSI work. In 1986, ISO and ANSI compromised on unified standard, the first official step in the IRDS becoming an International Standard.

However in 1987, ISO adopted British proposal and diverged from ANSI. Meanwhile ANSI, well advanced in the approval process and under great pressure to finalize IRDS, continued as before. The ANSI standard was finally approved in 1988, eight years after it was started.

What is an IRDS?

An IRDS is a software system that conforms to the ANSI or ISO standard for data dictionary systems. The IRDS standard represents international efforts to provide quality data dictionary system support for information engineering and management. The IRDS data dictionary systems have been designed to support the complexities of metadata, or data about your information, application systems, hardware, users, documentation and so forth.

An Information Resource Dictionary (IRD) is an application of the IRDS. An IRD is a highly structured type of database that can be used to design, monitor, control, describe, locate, protect, document data in information systems. An IRD can also be used to support a wide variety of functions such as CASE, Data Administration, and other system life cycle information management.

Purpose of the IRDS Standard

The purpose of the IRDS standard is to provide an organization's data dictionary system users with useful, flexible, and user-friendly data dictionary system software products to support all phases of the system life cycle. The standards specify that these products must share a common set of features independent of implementation. These standardized IRDS features are specified in terms of core and optional modules that allow vendors to implement IRDS features modularly and incrementally in their products. The modular structure of the IRDS permits user organizations to acquire only needed modules.

IRDS Benefits

Both government and industry benefit with the incorporation of the IRDS specifications into commercial data dictionary system software products. Due to the common set of features shared by IRDS products, organizations with multiple IRDS products from different vendors can increase user efficiency and improve metadata transportability from system to system. IRDS standards are designed to improve the quality of data dictionary system software, by giving users schema extensibility and life cycle support, and by giving vendors a common basis from which to work.

The Architecture of the IRDS

The ANSI and ISO IRDS Database can be viewed as a four-level architecture in which the information specified at one level describes (and potentially controls) the information stored at the next lower level. Thus one level defines the types of "objects" which can be described at the next lower level, and that level contains the "instances" of those objects. These four levels are illustrated in Figure B-1

Level 1 is fixed by the IRDS Standards Committees which specify the level 1 model.

Level 2 is the fundamental data model used to design level 3. This level is populated and maintained by corporate dictionary administrator and CASE vendors - the people who develop and implement methodologies for software engineering.

Level 3 describes application systems, processes, hardware, and users. Level 3 is the metadata and is primarily the concern of application developers, system analysts, database administrators and such.

At level 4, the lowest level, is end-user data. A dictionary does not actually contain any level 4 data; populating and maintaining this level is the business of the end users working with a DBMS.

The IRDS Standards Committees also provide edit, control, and reporting functions at level 2 and 3. IRDS Standards do not address level 4.

Figure B-1 The Four-Level Architecture of the IRDS

FEATURES OF THE ANSI IRDS

The ANSI IRDS Data Model

Entity-Relationship-Attribute Modeling

A schema is the part of a dictionary or database that defines the basis structures to be supported in that dictionary or database. The IRD Schema Descriptors provide the foundation on which metadata structure can be built. The IRD Schema Description is based on the Entity-Relationship information model which support the analysis of entities, relationships, and attributes.

The data dictionary contains information about data entities, the relationship between those entities, and their attributes. An IRD entity describes or represents a concept, event, person, or quantity, but it is not the actual data that exists in an application file or database. For example, an IRD entity may be a Social Insurance Number (SIN). The entity would not be the actual SIN "123-456-789". An IRD relationship would be an association between two IRD entities (e.g. payroll-record "CONTAINS" SIN). "CONTAIN" is the relationship between payroll-record and SIN entities. Attributes represent properties of an entity or relationship. In the above example, one attribute for entity SIN could be LENGTH. Figure B-2 [GOLD 87] depicts a small subset of an IRD, illustrating the organization of data in the IRD.

The IRDS recognizes only binary relationships, which associate two entities. The IRDS does not recognize ternary relationships.

Furthermore, at present, the relationships represented in an IRD are unidirectional, not reciprocal. For example, in the relationship-type Payroll-System-CONTAINS-Payroll-File, the IRDS does not automatically define or maintain a reverse relationship-type Payroll-File-IS-CONTAINED-IN-Payroll-System. As a result, the reverse relationship-type may be defined separately. Note that, it would be the IRD administrators responsibility to maintain the system integrity when it comes to avoiding inconsistencies between relationship-types and their associated reverse relationship-types.

Figure B-2 Subset of an information Resource Dictionary

The ANSI IRDS Architecture

The ANSI IRDS is a structured framework for organizing metadata information using a four layer architecture, which supports multiple levels of abstraction in terms of an entity-relationship model as discussed in previous section.

The four layered structures are:

- IRD Schema Description Level
- IRD Schema Level
- IRD Data Level
- Production Data Level

Each level in the framework has both a structure and an extent (collection of operational metadata) associated with it. The content of each level serves as a schema for the next lower level. That is, the content of each level collectively describes and controls the structure and content of the next lower level.

IRD Schema Description Level

This is the top level and is defined by the IRDS implementor. This level contains the types of all the objects (meta-entity-types) can be defined the the next lower level. the IRD Schema, the types of the relationships (meta-relationship-types) that can exist between these types

of objects, and certain properties (meta-attribute-types) of both of these types.

Metadata at the IRDS Schema Description level is completely specified by the IRDS Standard and cannot be modified. The metadata at this level can, however, be extended during IRDS implementation to more accurately represent metadata in an extended IRD Schema.

IRD Schema Level

This is the second level from the top and is the most important level of the IRDS standard. This level defines the types of objects that may be stored in the IRD. It also defines various control mechanisms, including naming rules, defaults, and validation information for the IRD contents.

IRD Data Level

This is the third level from the top and it describes the objects in the environment and the associations among those objects; the object descriptions are called entities, and the association descriptions are called relationships. This level also describes the properties of the objects and their associations; these properties are called attributes.

Production Data Level

This is the lowest level and is not described in the IRDS standard.

The IRDS Contents

The ANSI IRDS consists of seven modules:

- the **Core**, which defines the building blocks of the IRDS itself.
- the **Basic Functional IRD Schema**, which defines the content of the basic, or core, data dictionary.
- **IRDS Security**, which defines the required features for controlling access to the IRDS.
- the **Extensible Life Cycle Phase Facility**, which provides for management of the contents of the IRDS in terms of the system life cycle.
- the **Procedure Facility**, which provides for the creation of procedures for accessing and manipulating the content of the IRDS.
- the **Application Program Interface**, which provides for access to the IRDS from an application program written in a conventional programming language.
- **Entity Lists**, which provides for the creation and manipulating of named lists of IRDS entities.

The IRDS Core

The IRDS Core Module consists of the IRD Schema, the Minimal IRD Schema, and either the Command Language Interface or the Panel Interface, or both.

The IRDS Standard defines a core set of metadata necessary to describe and insure the integrity of the IRDS contents and to support extensibility. The core schema is required in all IRDS implementations and is made up of a collection of standard metadata at the IRD Schema Description level and another collection at the IRD Schema level (the **Minimal Schema**). Moreover, the core module also consists of either the **Command Language Interface** or the **Panel Interface**, or both. In addition, the Standard includes six optional modules, also at the IRD Schema level, to provide additional functionality outside of the required core IRDS. These six modules are: the **Basic Functional IRD Schema**, **IRDS Security Facility**, **Extensible Life Cycle Phase Facility**, **Procedure Facility**, **Application Program Interface**, and **Entity List**.

The IRDS supports an extensible schema that permits users to define and modify IRD schema structures. Predefined and user-defined schema structures are integrated in the IRDS. The extensible schema capability

provides the user with the flexibility to design an IRD schema to fit the particular metadata requirements of an organization of life cycle phase.

There are also other facilities and interfaces. Some of these are briefly discussed in the following sections. For detail information, refer to [GOLD 87] and [ANSI 88].

Minimal Schema

The IRDS standard specifies a Minimal Schema for every IRDS implementation. The IRDS minimal schema contains the collection of metadata (such as **meta-entities**, **meta-relationships**, **meta-attributes**, **meta-attribute-groups**) to establish controls over and insure the integrity of the IRD Schema and IRD. It is made up of metadata at the IRD Schema level. The minimal schema provides support for tracking update activity, maintain user views of the IRDS, validating attribute values, controlling movement of metadata through life cycle phases, and maintaining lists of default values and limits, reserved entity and meta-entity names, and rules for naming entities and meta-entities. The followings are the critical schema descriptors:

- **Entity-Types:** IRDS-USER, IRD-VIEW, and IRD-SCHEMA-VIEW
- **Attribute-Types:** ADDED-BY, IRD-PARTITION-NAME, LAST-MODIFIED-BY, IRD-SCHEMA-PHASE-NAME, SYSTEM-DATE, SYSTEM-TIME, and VALUE-VALIDATION
- **Attribute-Group-Type:** DATE-TIME-ADDED
- **Relationship-Class-Type:** HAS
- **Relationship-Types:** IRDS-USER-HAS-IRD-VIEW, IRDS-USER-HAS-IRD-SCHEMA-VIEW

Basic Functional Schema

The Basic Functional Schema, one of the IRDS modules, provides an initial set of schema structures that the user can use and build upon.

The main schema descriptors specified in this module are a set of **entity-types**, **attribute-types**, and **attribute-group-types**, which are described below.

The entity-types predefined in this module are grouped into three categories: **DATA**, **PROCESS**, and **EXTERNAL**.

DATA Entity-Types

- **DOCUMENT**, describing instances of human readable data collections.
- **FILE**, describing instances of an organizational's data collections.
- **RECORD**, describing instances of logically associated data that belong to an organization.
- **ELEMENT**, describing instances of data belonging to an organization.

PROCESS Entity-Types

- **SYSTEM**, describing instances of collections of data and processes.
- **PROGRAM**, describing instances of automated processes.
- **MODULE**, describing instances of automated processes that are called by **PROGRAM** entities.

EXTERNAL Entity-types

- **USER**, describing individuals or organizational components

Predefined relationship-class-types in the Basic Functional Schema

includes:

- **CONTAINS**, describing instances of one entity being composed of other entities.
- **PROCESSES**, describing associations between **PROCESS** and **DATA** entities.
- **RESPONSIBLE-FOR**, describing associations between entities representing organizational components and other entities, to denote organizational responsibility.
- **RUNS**, describing associations between **USER** and **PROCESS** entities.
- **GOES-TO**, describing data flow associations between entities.
- **DERIVED-FROM**, describing associations between entities where the target entity is the result of a calculation involving the source entity.
- **CALLS**, describing calling associations between **PROCESS** entities

Specific **relationship-types** between entity-types are also predefined for the user, such as: SYSTEM-CONTAINS-PP, DOCUMENT-CONTAINS-ELEMENT, PROGRAM-PROCESS-DOCUMENT, MODULE-PROCESSES-ELEMENT, USER-RESPONSIBLE-FOR-DOCUMENT, USER-RUNS-PROGRAM, PROGRAM-GOES-TO-PROGRAM, etc. The directionality of the required relationship-types is also predefined.

For the entity-types named above, many **attribute-types** are predefined for the user by the Basic Functional Schema, such as: ADDED-BY, ALLOWABLE-RANGE, CLASSIFICATION, LENGTH, DATE-TIME-ADDED, DESCRIPTION, EXTERNAL-SECURITY, LOCATION, and USAGE. A few attribute-types for relationship-types are also defined such as: FREQUENCY, which is associated with the relationship-types of PROCESSES and RUNS.

ANSI IRDS Facilities

IRDS Security Facilities

The Security Facilities allow organizations to restrict access to IRD and IRD Schema functionality and content. That is to support the restrictions of user access permissions to an IRD, to an IRD Schema, to an entity-type, to individual commands, and to individual entities. Two levels of IRD access control are provided:

- **Global Security**, which provides restriction mechanism to protect the IRD and IRD Schema. It controls user access restrictions according to entity-type, meta-entity-type, and partitions.
- **Entity-Level Security**, which provides the ability to assign read or write privileges for individual entities.

Extensible Life Cycle Phase Facility

This module extends the life-cycle-phase facilities of the Core IRDS by implementing integrity rules and customization features needed to control the movement of entities through the life-cycle. The structure of the life-cycle itself can be customized to match the organization's preferred methodology. Based on this structure, entities can be moved from UNCONTROLLED to CONTROLLED phases, from CONTROLLED to ARCHIVED phases, and from CONTROLLED to UNCONTROLLED

phases, with the organization retaining full control over the integrity constraints that regulate this movement.

Procedure Facility

This module provides the ability to define, store, maintain, and execute procedures involving the IRD and the IRD Schema. The operation of this module requires the presence of the Command Language interface.

Entity-List Facility

The IRDS allows a user to create and manipulate lists of access-name based on user-specified selection criteria. These entity-lists may then be input to other IRDS output functions and certain maintenance functions. The entities contained in an entity-list are always a subset of those contained in IRD-views specified by and authorized for the user.

ANSI IRDS Interfaces

Command Language and Panel Interfaces

The Command Language Interface supports the user's interaction with the IRDS in both batch and interactive modes. The Command Language is fully described in [GOLD 88c].

The Panel Interface provides a set of panels (i.e. conceptual screens) through which the user can access and manipulate an IRD. The Panel Interface is specified in terms of functional characteristics without definition of window or screen design. Each IRDS panel provides six information areas for the user: State Area, Data Area, IRD Schema Area, Action Area, Message Area, Help Area.

Application Program Interface

The Application Program Interface permits standard programming languages to interface with the command language of the IRDS. The Call feature of a standard language, such as COBOL, FORTRAN, C, or PL/I, can be used to access the metadata in an IRD. The IRDS standard does not specify particular language bindings. With this module, users can write program to collect metadata from, and pass metadata to, an IRD.

IRD-IRD Interface

The IRDS standard provides general specifications for an IRD-IRD Interface that supports schema and metadata interchange between separate IRDs, which may be located in one or separate IRDSs. The IRD-IRD Interface facility permits an organization using standard IRDSs to select and transport part or all of an IRD schema, and part or all of the corresponding IRD data, from one IRD to another IRD.

Since the IRD-IRD Interface is defined in the IRDS Core, every standard IRDS implementation has this capability.

APPENDIX C -- Overview of ISO IRDS

4

of/de

4

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

PIONEERS IN METHYLENE BLUE TESTING SINCE 1974

MICROD
SOLUTIONS

1040 COUNTY ROAD 5 BURNINGWELL WY SUITE 100
TEL: 812-425-7887 FAX: 812-425-7887 TULSA, OK 74106

Contents

Overview of ISO IRDS

Fundamental Level
IRD Definition Level
IRD Level
Application Level
Level Pairs
IRDS Facilities
IRDS Interfaces

Overview of ISO IRDS

The ISO IRDS functionality is associated with three data levels, but it is useful for expository reasons to cover four data levels.

The four data levels described are as follows:

- Fundamental Level;
- IRD Definition Level;
- IRD Level;
- Application Level.

These levels, while being inter-related, exist to serve different purposes, as will be described later.

Fundamental Level

The purpose of the Fundamental level is to prescribe the types of object about which data may be recorded on the IRD Definition Level.

The definition of the types of data that can be stored on the IRD Definition Level is called the IRD Definition Schema. It is necessary to use a data modelling approach to define the IRD Definition Schema. This data modelling approach may be one defined by a database language standard.

IRD Definition Level

The purpose of the IRD Definition Level is to contain IRD definitions. The types of data whose instances are recorded in IRD Definitions are defined on the Fundamental Level. There may be any number of IRD Definitions existing, all described by one IRD Definition Schema.

A part of an IRD Definition, referred to as an IRD Schema, prescribes the types of object about which data may be stored in one or more IRDs.

An IRD Definition may contain one or more IRD Schemas. Some of the content of the IRD Definition may be under development and intended to replace the content of an IRD Schema or to add to the definitions in an IRD Schema.

An IRD Definition includes IRD control information.

IRD Level

The purpose of the IRD Level is to contain IRDs. There may be any number of IRDs existing, all described by one IRD Schema. There may be also be other IRDs described by other IRD Schemas.

Some, but not all, of the content of an IRD defines types at the application level.

For example, an IRD would contain the information that EMPLOYEE and PURCHASE ORDER are two record types. An IRD might also contain information about which programs use these record types.

The purpose of the data in an IRD is to enable the IRDS to support the design, construction and operation of computerized information systems, and any other functions for which the IRDS is seen as an appropriate tool.

An IRD Definition specifies further types of data in an IRD, necessary for day to day administration by an dictionary administrator.

Some of the data instances in an IRD may be defined in a Standard. Other content of an IRD may be added.

There is no restriction on the types of data which may be held on the IRD level, provided that the instances of this data correspond to types in the applicable IRD Schema.

Application Level

The Application Level is the level on which instances of business data are recorded.

Level Pairs

The ISO provides services at two levels: the IRD Definition Level and the IRD Level. In each case the services may be thought of as operating on a level pair. A level pair consists of two adjacent data levels. The upper level of the two will always contain the "type" of information relevant to the "instances" on the lower level of the two.

The successful operation of an ISO IRDS requires the data on the lower level of any pair to be consistent with a specified version of its schema at the higher level of the level pair.

If a schema is changed, for which data exists at the lower level, consistency must be maintained between that data and the schema.

The three level pairs are identified as follows:

- IRD Definition Level Pair;
- IRD Level Pair;
- Application Level Pair.

Only the first two level pairs are the subject of the IRDS standards. Each service provides for in the IRDS Services Interfaces Standard relates to a level pair and never to a single level.

IRDS Facilities

The ISO IRDS facilities will provide users with the services to:

- insert, update, and delete data under its control;
- select and retrieve data;
- provide means of specifying constraints on the values associated with objects, and on the associations between objects;
- provide means of limiting access to the data and users;
- provide means of optionally auditing auditing changes to an IRD and IRD definition.
- allow the specifications of limit and default values of specified attributes;
- provide query and reporting capabilities
- provide remote data access
- provide version control;
- provide impact analysis.

IRDS Interfaces

The ISO also provide IRDS Panel Interface, IRDS Command Language Interface, Export and Import Interface, etc.

Appendix D - Detailed Design for IRD Layers

Contents

(A) Design for IRD Data Layer

- **Entities for IRD Data Layer**
- **Entity-Relationship Model for Prototype's IRD Data Layer**
- **Schema Descriptions for IRD Data Layer**
- **Schema Design for IRD Data Layer**

(B) Design for IRD Schema Layer

- **Entities for IRD Schema Layer**
- **Entity-Relationships Model for Prototype's IRD Schema Layer**
- **Schema Descriptions for IRD Schema Layer**
- **Schema Design for IRD Schema Layer**

(A) DESIGN FOR IRD DATA LAYER**Entities for IRD Data Layer**

The entities required to describe the prototype's IRD Data Layer are presented below:

APPLICATION. An application entity represents an application system. It has a name, description, etc. It can relate to PROGRAMs, EMPLOYEEs, APPLICATION_ACCOUNTs, DBMSs.

APPLICATION_ACCOUNT. This entity represents information for an APPLICATION's database accounts. It can relate to APPLICATION, TABLEs, INDEXes, DATABASE_ACCOUNT_PRIVILEGEs.

BACKUP_INFORMATION. This entity represents the database backup schedule and the status of the last backup. It can relate to DATABASE.

CPU. This entity represents computing device information. It can relate to OPERATING_SYSTEM, NETWORKs, DATABASEs.

DATABASE. This entity represents a database such as production database, development database, test database, etc. It can relate to CPU, DBMS, BACKUP_INFORMATION, SHUTDOWN_SCHEDULEs, SOFTWARE_TOOLS, DATABASE_ACCOUNTs, DATABASE_FILEs.

DATABASE_ACCOUNT. This entity represents database account privilege information. It can relate to DATABASE, TABLESPACEs, APPLICATION_ACCOUNT.

DATABASE_FILE. This entity represents physical files information. It can relate to DATABASE, TABLESPACE.

DBMS. This entity represents database management system information. It can relate to DATABASEs, APPLICATIONs.

EMPLOYEE. This entity represents an individual responsible for the applications. EMPLOYEEs will have names and descriptive information. It will be possible to relate an EMPLOYEE to many other entities.

However, it is only related to an APPLICATION in the prototype.

INDEX. This entity represents indexes for relation tables. It can relate to TABLE, APPLICATION_ACCOUNT, TABLESPACE.

NETWORK. This entity represents communication network information. It can relate to CPUs.

OPERATION_SYSTEM. This entity represents operating system information. It can relate to CPUs.

PROGRAM. This entity represents application programs for an application. It can relate to PROGRAMs, MODULEs.

SHUTDOWN_SCHEDULE. This entity represents database shutdown schedule information. It can relate to DATABASE.

SOFTWARE_TOOL. This entity represents software tools. It can relate to DATABASEs.

TABLE. This entity represents relation tables. It can relate to INDEXes, APPLICATION_ACCOUNT, TABLESPACE.

TABLESPACE. This entity represents DBMS logical file information. It can relate to INDEXes, TABLEs, DATABASE_ACCOUNT, DATABASE_FILEs.

Entity-Relationship Model for Prototype's IRD Data Layer

Figure D-1 Entity-Relationship Model for Prototype's IRD Data Layer

Notes: represents one to many relationship

Schema Descriptions for IRD Data Layer

<u>Table</u>	<u>Column Names</u>	<u>Descriptions</u>
T_APPLICATION	APPL_ID APPL_DESC STATUS	Application acronym Application description Status of the Application
T_APPL_DB_ACCT	DB_ID APPL_DB_ACCT APPL_ID FOR_VERSION	Database ID Appl DB Acct Appl acronym Appl of a version #
T_CPU	CPU_ID CPU_DESC OP_SYS OS_VER	Computer ID Description of computer Operating system (OS) Version of OS
T_DATABASE	DB_ID DB_DESC DBMS_TYPE RESIDE_IN_CPU DB_NAME DB_INST_ID DB_IMAGE_ID ACCESS_STRING	Database ID Database description Type of DBMS Resident CPU for the DB DB Name DB Instance ID DB Image ID Command to invoke the DB
T_DB_ACCT_PRIV	DB_ID DB_ACCT CONN_PRIV RESOURCE_PRIV DBA_PRIV DATE_EXTRACTED	Database ID DB Account ID CONNECT privilege RESOURCE privilege DBA privilege Extraction Date of row
T_DATABASE_FILE	DB_ID DBS_FILE SIZE_IN_BYTE STATUS TS_NAME DATE_EXTRACTED	Database ID Database file name Size of DBS file Status of DBS file Associated tablespace Extraction Date of row

<u>Table</u>	<u>Column Names</u>	<u>Description</u>
T_INDEX	DB_ID IDX_OWNER IDX_NAME TS_NAME TABLE_NAME INIT_EXT NEXT_EXT PCT_INCR ALLOC_BYTE ALLOC_EXT DATE_EXTRACTED	Database ID Owner of index Index name Tablespace of index Name of table Initial extent Next extent % increase of allocated bytes of allocated segments Extraction Date of row
T_NETWORK	NET_ID NET_DESC NET_TYPE	Company network ID Network description Type of network
T_TABLE	DB_ID TABLE_OWNER TABLE_NAME TS_NAME INIT_EXT NEXT_EXT PCT_INCR ALLOC_BYTE USED_BYTE ALLOC_EXT DATE_EXTRACTED	Database ID Owner of table Table name TS for table Initial extent Next extent % increase Allocated bytes Used bytes # of allocated segments Extraction Date of row
T_TABLESPACE	DB_ID TS_NAME FREE_BYTE N_EXTENT STATUS DATE_EXTRACTED	Database ID Tablespace name Free space in bytes # of extents Tablespace status Extraction Date of row
T_CPU_IN_NET	CPU_ID CONN_TO_NET	Computer ID Network ID
T_ACCT QUOTA	DB_ID DB_ACCT TS_NAME TS_QUOTA USED_QUOTA DATE_EXTRACTED	Database ID DB Account ID Assigned TS Assigned TS quota (bytes) Quota used by the Acct. Extraction Date of row
OPSYS ENT	OS_TYPE OS_VERSION VENDOR	Operating system Version Vendor of Op.Sys.

<u>Table</u>	<u>Column Names</u>	<u>Description</u>
DBMS_ENT	DBMS_NAME VERSION DBMS_DESC	Name of the DBMS Version of the DBMS Description of DBMS
ELE_ENT	ELE_NAME ELE_DESC ALIAS_OF	Element Name Element Description Alias of Element
OPER_ENT	OPER_TYPE OPER_DESC REF_TBL_NAME	Operation Type Operation Description For future use
PGM_ENT	PGM_NAME OWNED_BY_SYS PGM_DESC PGM_EXT LOCATION WRITTEN_IN FOR_DBMS ADDED_BY DATE_ADDED MODIFIED_BY DATE_MODIFIED	Program Name Application Owner Program Description Program file extension Directory location of the Program Language used by the program Written for DBMS User who added the program Program added date Program modified by User Program was last modified
T_DB_SHUTDOWN_ SCHEDULE	SHUTDOWN# DB_ID NOTE FREQUENCY FROM_DAY FROM_TIME TO_DAY TO_TIME	Shutdown ID Database ID Information node Shutdown frequency Day on which Database shutdown Shutdown time for the specify Day Ending shutdown day Ending shutdown time
SW_ENT	SW_ID SW_NAME VERSION	Software ID Software name Software version
APPL_CONTACTS	APPL_ID EMP_NUM	Application ID Employee responsible for the application
DB_CONT_SW	DB_ID SW_ID	Database ID Software ID
R_SYS_USES_DBMS	SYS_NAME DBMS_NAME	Application ID DBMS ID

Schema Design for IRD Data Layer

(I) Schema Design for other IRD Data Layer Entities:

APPLICATION, APPLICATION ACCOUNT, BACKUP_INFO, CPU, DATABASE, DBMS, EMPLOYEE, NETWORK, OPERATING SYSTEM, SHUTDOWN SCHEDULE, SOFTWARE TOOL and the related Relationships:

Entities

```
create table T_APPLICATION
(appl_id          char(6)  primary key,
 appl_desc       char(40),
 status          char(1),
 category        char(10));

create table T_APPL_DB_ACCT
(db_id           number(3)  not null,
 appl_db_acct    char(15)  not null,
 appl_id         char(6)   not null,
 for_version     number,
 primary key (db_id, appl_db_acct, appl_id),
 foreign key (db_id,appl_db_acct) references
            mdas.t_db_acct_priv (db_id,db_acct),
 foreign key (appl_id) references mdas.t_application (appl_id));

create table T_DB_BACKUP_INFO
(backup#         number  primary key,
 job_name        char(20),
 job_desc        char(30),
 start_time      char(5),
 frequency       char(7),
 db_id           number,
 last_backup_on  date,
 last_backup_status char(7),
 foreign key (db_id) references mdas.t_database (db_id));

create table T_CPU
(cpu_id          char(4)   primary key,
 cpu_desc        char(30),
 op_sys          char(10),
 os_ver          char(10));
```

```
create table DBMS_ENT
```

```
(dbms_name      char(6) primary key,
 version        char(8),
 dbms_desc      char(20));
```

```
create table T_DATABASE
```

```
(db_id          number(3) primary key,
 db_desc        char(20),
 dbms_type      char(6),
 reside_in_cpu  char(4),
 db_name        char(8),
 db_inst_id     char(6),
 db_image_id    char(35),
 foreign key (dbms_type) references mdas.dbms_ent (dbms_name),
 foreign key (reside_in_cpu) references mdas.t_cpu (cpu_id));
```

DESC SH_EMPLOYEE (Public Table)

<u>Name</u>	<u>Null?</u>	<u>Type</u>
EMPLOYEE_NUMBER	NOT NULL	CHAR(5)
INITIALS		CHAR(3)
SURNAME		CHAR(20)
WORKS_IN_BRANCH		CHAR(4)
SITE_LOCATED		CHAR(2)
BUILDING_LOCATED		CHAR(8)
MAILING_STATION		CHAR(8)
TELEPHONE_DN		CHAR(15)
GIVEN_NAME		CHAR(12)
DIVISION_CODE		CHAR(2)
PAID_BY_BRANCH		CHAR(4)
TERMINATION_DATE		DATE

```
create table T_NETWORK
```

```
(net_id        char(6)      primary key,
 net_desc      char(30),
 net_type      char(10));
```



```

create table PGM_ENT
(pgm_name          char(30),
 owned_by_sys      char(6),
 pgm_desc          char(50),
 pgm_ext           char(6),
 location          char(40),
 written_in        char(10),
 for_dbms          char(6),
 added_by          char(20),
 date_added        date,
 modified_by       char(20),
 date_modified     date,
primary key (owned_by_sys, pgm_name),
foreign key (owned_by_sys) references mdas.T_APPLICATION
(appl_id),
foreign key (for_dbms)      references mdas.DBMS_ENT
(dbms_name));

```

rem.. This table contains the occurrences of the
rem.. OPERATING SYSTEM Entity
rem..

```

create table OPSYS_ENT
(os_type          char(10),
 os_version       char(8),
 os_vendor        char(10),
primary key (os_type, os_version, os_vendor));

create table T_DB_SHUTDOWN_SCHEDULE
(shutdown#        number primary key,
 db_id            number,
 note             char(30),
 frequency        char(7),
 from_day         char(3),
 from_time        char(5),
 to_day           char(3),
 to_time          char(5),
foreign key (db_id) references mdas.t_database (db_id));

```

```

create table SW_ENT
(sw_id            number primary key,
 sw_name          char(20),
 version          char(8));

```

Relationships

```
create table APPL_CONTACTS
(appl_id          char(6) not null,
 emp_num         char(5) not null,
primary key (appl_id, emp_num),
foreign key (appl_id) references mdas.t_application (appl_id));
```

```
create table T_CPU_IN_NET
(cpu_id          char(4) not null,
 conn_to_net     char(6) not null,
primary key (cpu_id, conn_to_net),
foreign key (cpu_id) references mdas.t_cpu (cpu_id),
foreign key (conn_to_net) references mdas.t_network (net_id));
```

```
create table DB_CONT_SW
(db_id          number not null,
 sw_id         number not null,
primary key (db_id, sw_id),
foreign key (db_id) references mdas.t_database (db_id),
foreign key (sw_id) references mdas.sw_ent (sw_id));
```

```
create table R_SYS_USES_DBMS
(sys_name       char(6) not null,
 dbms_name      char(6) not null,
primary key (sys_name, dbms_name),
foreign key (sys_name) references mdas.t_application (appl_id),
foreign key (dbms_name) references mdas.dbms_ent (dbms_name));
```

(II) Schema Design for other IRD Data Layer Entities and Relationships that are required to support the prototype IRDS implementation:

Entities

rem.. This table contains the occurrences of the
rem.. ACCOUNT Entity

rem..

create table ACCT_ENT

(acct_type char(15) primary key,
acct_desc char(30),
ref_tbl_name char(25));

rem..

rem.. This table contains the occurrences of the
rem.. ELEMENT Entity

rem..

create table ELE_ENT

(ele_name char(20) primary key,
ele_desc char(30),
alias_of char(20));

rem.. This table contains the occurrences of the
rem.. OPERATION Entity

rem..

create table OPER_ENT

(oper_type char(15) primary key,
oper_desc char(30),
ref_tbl_name char(25));

Relationships

create table DB_CONT_OPER

(db_id number not null,
oper_type number not null,
primary key (db_id, oper_type),
foreign key (db_id) references mdas.t_database (db_id),
foreign key (oper_type) references mdas.oper_ent (oper_type));

create view V_DB_CONT_OPER (dbid,oper,freq) as

select db_id, 'SHUTDOWN', frequency
from mdas.t_db_shutdown_schedule

UNION

select db_id, 'BACKUP', frequency
from mdas.t_db_backup_info;

(III) Schema Design for Captured System Catalog Data (i.e. for entities: DATABASE ACCOUNT, DATABASE FILE, INDEX, TABLE, TABLESPACE and the related relationship):

Entities

```
create table T_DATABASE_FILE
(db_id          number(3)  not null,
 db_file        char(20)  not null,
 size_in_byte   number,
 status         char(9),
 ts_name        char(15),
 date_extracted date,
 primary key (db_id, db_file),
 foreign key (db_id) references mdas.t_database (db_id));
```

```
create table T_DB_ACCT_PRIV
(db_id          number(3)  not null,
 db_acct        char(20)  not null,
 conn_priv      number,
 resource_priv  number,
 dba_priv       number,
 date_extracted date,
 primary key (db_id, db_acct));
```

```
create table T_INDEX
(db_id          number(3)  not null,
 idx_owner      char(15)  not null,
 idx_name       char(30)  not null,
 ts_name        char(15),
 table_name     char(30),
 init_ext       number,
 next_ext       number,
 pct_incr       number,
 alloc_byte     number,
 alloc_ext      number,
 date_extracted date,
 primary key (db_id, idx_owner, idx_name),
 foreign key (db_id) references mdas.t_database (db_id));
```

```

create table T_TABLE
(db_id          number(3) not null,
 table_owner    char(15) not null,
 table_name     char(30) not null,
 ts_name        char(15),
 init_ext       number,
 next_ext       number,
 pct_incr       number,
 alloc_byte     number,
 used_byte      number,
 alloc_ext      number,
 date_extracted date,
 primary key (db_id, table_owner, table_name),
 foreign key (db_id) references mdas.t_database (db_id));

```

```

create table T_TABLESPACE
(db_id          number(3) not null,
 ts_name        char(15) not null,
 free_byte      number,
 n_extent       number,
 status         char(9),
 date_extracted date,
 primary key (db_id, ts_name),
 foreign key (db_id) references mdas.t_database (db_id));

```

Relationship

```

create table T_ACCT_QUOTA
(db_id          number(3) not null,
 db_acct        char(20) not null,
 ts_name        char(15),
 ts_quota       number,
 used_quota     number,
 date_extracted date,
 primary key (db_id, db_acct, ts_name),
 foreign key (db_id, db_acct) references mdas.t_db_acct_priv
(db_id,db_acct),
 foreign key (db_id, ts_name) references mdas.t_tablespace
(db_id,ts_name));

```

(B) DESIGN FOR IRD SCHEMA LAYER**Entities for IRD Schema Layer**

The meta-entity descriptions for the IRD Schema Layer are discussed below:

ENT_TYPE (ENTITY_TYPE) This meta-entity contains the types of entities that are stored in the IRD Data Layer. Examples: SYSTEM, USER, DATABASE, DBMS.

ATT_TYPE (ATTBIBUTE_TYPE) This meta-entity contains the types of attributes that are used to describe the entity-types in the IRD Data Layer. Examples: LENGTH, ACRONYM, LAST_MODIFIED.

REL_CLASS_TYPE (RELATIONSHIP_CLASS_TYPE) This meta-entity contains the types of relationship that are used to express the acts, occurrences, or modes that connect entity-types in the IRD Data Layer. Examples: CONTAINS, LINKS, OWNS, CALLS, USES.

ENT_REL (ENTITY_RELATIONSHIP) This table contains the entity-types that are linked by the Relationship Class Types.

V_REL_TYPE (RELATIONSHIP_TYPE) This meta-entity contains the allowable connections between entity-types in the IRD Data Layer. It is implemented as a view on the ENT_REL table. Examples: SYSTEM-USES-DBMS.

ENT_TYP_CONTAINS_ATT_TYP

(ENTITY_TYPE_CONTAINS_ATTRIBUTE_TYPE) This meta-relationship contains attributes-types for the entity-types.

REL_TYP_CONTAINS_ATT_TYP**(RELATIONSHIP_TYPE_CONTAINS_ATTRIBUTE_TYPE)**

This meta-relationship contains attribute-types for the relationship-types.

These tables support the basic IRDS concepts of entities and relationships, with both entities and relationships having attributes. In IRDS, relationships are restricted to relating entities to entities, not entities to relationships or relationships to relationships. The current IRDS standard supports only binary relationships, not the general n-ary relationships provided by some entity-relationship models. The tables at the IRD schema level support these concepts by providing several tables in which new objects and their attributes are recorded. (At the IRD schema layer, entries in these tables are called "meta-entities" or "meta-relationships") [CSR 88]. The ENTITY_TYPE table allows new kind of objects, e.g. a "Database", to be created. The attributes of the new object, e.g. Name of Database, are assembled using the "ENT_TYP_CONTAINS_ATT_TYP" relationship. Each attribute of the new object is an instance in this relationship. After a new object is

defined, the IRDS can record instances of this object, e.g., "1", "PROD", "ORACLE", and so forth. Before an attribute can be associated with an object, the attribute must be defined in the ATTRIBUTE_TYPE table.

This allows the characteristics of the attribute itself to be specified.

Each attribute can be associated with as many entities or relationships as desired.

To create a new relationship, an additional step is required to declare the entities which are connected by the relationship; this information is entered into the "ENT_REL" table which contains the name of the relationship class type and both of the entity types which are connected by the relationship class type. The "RELATIONSHIP_TYPE" table is then implemented as view of this underlying table. Attributes to specify the relationship can be recorded in the "REL_TYP_CONTAINS_ATT_TYP" table.

Entity-Relationship Model for Prototype's IRD Schema Layer **Figure D-2 Entity-Relationship Model for IRD Schema Layer**

Schema Descriptions for IRD Schema Layer

<u>Table</u>	<u>Column</u>	<u>Descriptions</u>
ENT_TYPE	ENT_NAME	Short form of entity name
	ENT_DESC_NAME	Full descriptive name of entity
	ENT_DEFN	Entity definition
	ADDED_BY	User who added the entity
	DATE_ADDED	Date or which the entity was added
	MODIFIED_BY	User who modified the entity
	DATE_MODIFIED	Date or which the entity was modified
	REF_TBL_NAME	Table name of reference table (future use)
ATT_TYPE	ATT_NAME	Attribute name's short form
	ATT_DESC_NAME	Full attribute name
	ATT_DEFN	Attribute definition
	ADDED_BY	User who added the entity
	DATE_ADDED	Date or which the entity was added
	MODIFIED_BY	User who modified the entity
	DATE_MODIFIED	Date or which the entity was modified
REL_CLASS_TYPE	REL_CL_NAME	Relationship class's short name
	REL_CL_DESC_NAME	Relationship class's full name
	REL_CL_DEFN	Definition for rel. class
	ADDED_BY	User who added the entity
	DATE_ADDED	Date or which the entity was added
	MODIFIED_BY	User who modified the entity
	DATE_MODIFIED	Date or which the entity was modified
ENT_REL	ENT1	First entity
	REL_CL_TYPE	Relationship class type
	ENT2	Second entity
	REL_DESC	Rel. Type Desc Name
	ADDED_BY	User who added the entity
	DATE_ADDED	Date or which the entity was added
	MODIFIED_BY	User who modified the entity
	DATE_MODIFIED	Date or which the entity was modified
	REF_TBL_NAME	Table name of reference table (future use)

<u>Table</u>	<u>Columns</u>	<u>Descriptions</u>
ENT_TYP_CONTAINS_ATT_TYP	ENT_NAME	Entity type name
	ATT_NAME	Attribute type name
REL_TYP_CONTAINS_ATT_TYP	REL_NAME	Relationship Type name
	ATT_NAME	Attribute type name

Schema Design for IRD Schema Layer

Schema Design for IRD Schema Layer

rem..

rem.. Schema Definition for table ENTITY TYPE

rem..

create table ENT_TYPE

```
(ent_name          char(6) primary key,
 ent_desc_name     char(15),
 ent_defn          char(80),
 added_by          char(20),
 date_added        date,
 modified_by       char(20),
 date_modified     date,
 ref_tbl_name      char(30));
```

rem..

rem.. Schema Definition for table ATTRIBUTE TYPE

rem..

create table ATT_TYPE

```
(att_name          char(15) primary key,
 att_desc_name     char(30),
 att_defn          char(80),
 added_by          char(20),
 date_added        date,
 modified_by       char(20),
 date_modified     date);
```

rem..

rem.. Schema definition for

rem.. table RELATIONSHIP CLASS TYPE

rem..

create table REL_CLASS_TYPE

```
(rel_cl_name       char(6) primary key,
 rel_cl_desc_name  char(15),
 rel_cl_defn       char(80),
 added_by          char(20),
 date_added        date,
 modified_by       char(20),
 date_modified     date);
```

```

rem..
rem..  Schema Definition for table ENT_REL which
rem..  links entity types to the rel. class type.
rem..
create table ENT_REL
(ent1                char(6) not null,
 rel_cl_name        char(6) not null,
 ent2                char(6) not null,
 rel_desc           char(40),
 added_by           char(20),
 date_added         date,
 modified_by        char(20),
 date_modified      date,
 ref_tbl_name       char(30),
 primary key (ent1, rel_cl_name, ent2),
 foreign key (ent1) references mdas.ent_type (ent_name),
 foreign key (ent2) references mdas.ent_type (ent_name),
 foreign key (rel_cl_name) references mdas.rel_class_type
(rel_cl_name));

create view V_REL_TYPE (rel_name) as
  select ent1 || '-' || rel_cl_name || '-' || ent2
  from mdas.ENT_REL;

create table ENT_TYP_CONTAINS_ATT_TYP
(ent_name            char(6) not null,
 att_name            char(15) not null,
 primary key (ent_name, att_name),
 foreign key (ent_name) references mdas.ent_type (ent_name),
 foreign key (att_name) references mdas.att_type (att_name));

rem..
rem..  Schema Definition for table
rem..  RELATIONSHIP TYPE -CONTAINS- ATTRIBUTE TYPE
rem..
create table REL_TYP_CONTAINS_ATT_TYP
(rel_name            char(20) not null,
 att_name            char(15) not null,
 primary key (rel_name, att_name),
 foreign key (att_name) references mdas.att_type (att_name));

```

Appendix E - Sample Programs, Screens and Output

Listings

Note: Due to the volume of the prototype program listings, only few pages of each appendix are attached.

Appendix E1 - MDAS Interface Program

- The Interface Program (MDAS.MNU)
- Sample Printouts of Menu Displays

Note: Only the first item of the above is attached to this Appendix.

The Interface Program (MDAS.MNU)**Full Menu Application Information****A.1 Application Information**

Application:	MDAS
Short Nmae:	MDAS
Short Name:	MDAS
File Name:	MDAS
Creation Date:	31-MAR-91
Creator:	OPS\$CHEUNGD
Last Release Date:	
Menu Directory:	AECL\$MCB:[CHEUNGD.MDAS.MENU]
Version:	1
Identification:	MDAS

Full Menu Application Information

A.2 Menu overview

MDAS

MDAS_ADMIN

IRD_OPERATIONS

MAINT_IRD_DATA

MAINT_IRD_SCHEMA

Q_R_IRD_DATA

Q_R_IRD_SCHEMA

OPER_ON_DB

CAP_DB_DATA

Q_R_DATABASE_INFO

Note: In the next page, one of the above menu items is expanded and described in detail. Other menu items are not described further so as to limit the size of this Appendix

Full Menu Application Information**A.3 CAP_DB_DATA**

+-----+	
	Capture Database Data.
	Sub Menu
1	Capture Database File Data
2	Capture Tablespace Data
3	Capture Table Data
4	Capture Index Data
5	Capture Database Account Privileges Data
6	Capture Database Account Quota Data
	Purpose:
+-----+	

Note: Items 1 to 6 (i.e. A.3.1, ..., A.3.6) are described in detail in the next page.

A.3.1 Item Number: 1**Short Name:** Cap DBS File**Command Type:** Runform.**Command Line:**

=====

runform mdas\$scr:cep_extract /

Hint:

=====

No detailed help available for this item.

Item Roles:

=====

MDAS_ADMINISTRATOR

A.3.2 Item Number: 2**Short Name:** Cap TS Data**Command Type:** Runform.**Command Line:**

=====

runform mdas\$scr:cep_extract /

Hint:

=====

No detailed help available for this item.

Item Roles:

=====

MDAS_ADMINISTRATOR

A.3.3 Item Number: 3**Short Name:** Cap Table Data**Command Type:** Runform.**Command Line:**

=====

runform mdas\$scr:cep_extract /

Hint:

=====

No detailed help available for this item.

Item Roles:

=====

MDAS_ADMINISTRATOR

A.3.4 Item Number: 4

Short Name: Cap Index Data

Command Type: Runform.

Command Line:

=====

runform mdas\$scr:cep_extract /

Hint:

=====

No detailed help available for this item.

Item Roles:

=====

MDAS_ADMINISTRATOR

A.3.5 Item Number: 5

Short Name: Cap Acct Priv

Command Type: Runform.

Command Line:

=====

runform mdas\$scr:cep_extract /

Hint:

=====

No detailed help available for this item.

Item Roles:

=====

MDAS_ADMINISTRATOR

A.3.6 Item Number: 6

Short Name: Cap Acct Quota

Command Type: Runform.

Command Line:

=====

runform indas\$scr:cep_extract /

Hint:

=====

No detailed help available for this item.

Item Roles:

=====

MDAS_ADMINISTRATOR

Appendix E2 - Catalog Extraction Programs

- Tables for Catalog Extraction Process
- List of Catalog Extraction Programs
- Sample Printouts of CEP Menu and Screen Displays
- Catalog Extraction Program Listings
- CEP Screen Program Listing
- Sample Batch Job File created by CEP

Note: Only the first two items of the above are attached to this Appendix.

Tables for Catalog Extraction Process

The schema descriptions of the control (EXTR_CTL_INFO) and temporary (TEMP_CEP_PARM) tables of the Catalog Extraction Process are shown below:

<u>Table Name</u>	<u>Columns</u>	<u>Descriptions</u>
EXTR_CTL_INFO	DB_ID	Database ID
	APPL_DB_ID	Database ID of the application
	ADDED_ON	Record creation
	ADDED_BY	User added record
TEMP_CEP_PARM	TARGET_NODE	Target node
	TARGET_DB	Target database
	DBMS_TYPE	Type of DBMS
	FREQUENCY	For future use
	START_TIME	"
	EXTR_DBS	"
	EXTR_IDX	"
	EXTR_TBL	"
	EXTR_TBLSPACE	"
	EXTR_UPRIV	"
	EXTR_UQUOTA	"

List of Catalog Extraction Programs

A list of the Catalog Extraction Programs are shown below:

<u>Program Name</u>	<u>Description</u>
EXT_V5_DBS_FILE.SQL	To extract V5 Oracle database file info.
EXT_V6_DBS_FILE.SQL	To extract V6 Oracle database file info.
EXT_V5_DB_ACCT.SQL	To extract database account info.
EXT_V6_DB_ACCT.SQL	for V5/V6 Oracle database
EXT_V5_INDEX.SQL	To extract index information for the
EXT_V6_INDEX.SQL	relevant tables in T_TABLE
EXT_V5_ACCT_QUOTA.SQL	To extract acct quota information for
EXT_V6_ACCT_QUOTA.SQL	acct with Resource or DBA privilege
EXT_V5_TABLE.SQL	To extract tables with more than two
EXT_V6_TABLE.SQL	extents
EXT_V5_TABLESPACE.SQL	To extract partition (V5) or
EXT_V6_TABLESPACE.SQL	tablespace(V6) information

Other associated programs are presented below:

<u>Program Name</u>	<u>Description</u>
CR_CAT_EXTRACTION.SQL	To create command files that are used to extract metadata from the target database.
APPEND_COPY_SUBMIT.COM	To append the generated command files into a complete job file and then copy it to the target node for execution.
IMP_TO_IRD.COM	To invoke the proper command file to import the extracted metadata into the IRD.
IMP_ORA5.COM	To import the Oracle V5 metadata into the IRD.
IMP_ORA6.COM	To import the Oracle V6 meta data into the IRD.
IMP_V5_TO_MDAS.COM	To invoke export utility to load Oracle V5 metadata into the IRD.
CLEANUP.SQL	To delete the data from the IRD before loading new data.

Appendix E3 - Access Programs

- List of Access Programs
- Sample Printouts of Menu and Screen Display
- Report Program Listings
- Screen Program Listings
- Sample Report Outputs

Note: Only the first item of the above is attached to this Appendix.

List of Access Programs

A list of the Access Programs is given below:

Access Program

Function Description

Reports

ACCT_PRIV_REP.SQL	Report on Database Account Privilege Information.
ACCT_QUOTA_REP.SQL	Report on Database Account Quota Information.
DBS_FILE_REP.SQL	Report on Database File Information.
INDEX_REP.SQL	Report on Index Information for possible fragmented tables.
TABLE_REP.SQL	Report on possible fragmented Tables.
TABLESPACE_REP.SQL	Report on Tablespace Information.

Screens

DB_ACCT_PRIV.INP	Screen program for querying or reporting on Database Account Privilege Information.
ACCT_QUOTA.INP	Screen program for querying or reporting on Database Account Quota Information.
DBS_FILE.INP	Screen program for querying or reporting on Database File Information.
INDEX.INP	Screen program for querying or reporting on Index Information.
TABLE.INP	Screen program for querying or reporting on Fragmented Table Information.
TABLESPACE.INP	Screen program for querying or reporting on Tablespace Information.

Appendix E4 - Command Programs

- Table for Command Program
- List of Command Programs
- Sample Printouts of Menu and Screen Displays
- Command Program Listings
- Screen Program (ACCT_REG.INP) to Enroll User
- Sample Batch Job File Created

Note: Only the first two items of the above are attached to this
Appendix.

Table for Command Program

The schema description for the temporary table is given below:

<u>Table Name</u>	<u>Table Columns</u>	<u>Column Description</u>
TEMP_ACCT_REG	USER_ID	Database ID for the user
	USER_PW	Password for the account
	CPU	The target node
	DB_ID	The ID of the target database

List of Command Programs

A list of the Command Program is given below:

<u>Program Name</u>	<u>Description</u>
ACCT_REG.INP	Screen program to interface with the temporary table TEMP_ACCT_REG.
CR_USER_COM.SQL	To create a job file which contains commands to create the user accounts in the target production databases.
COPY_SUBMIT.COM	To copy the job file (created by the above program) to the target node and then submit it to the batch queue for execution.
ADD_ORA5_USER.SQL	Create end user account for Oracle RDBMS V5.
ADD_ORA6_USER.SQL	Create end user account for Oracle RDBMS V6.

Appendix E5 - IRD Operation Programs

- List of IRD Schema Layer Programs
- List of IRD Data Layer Programs
- Sample Printouts of IRD MENUS AND SCREENS
- Sample Program Listings for IRD Schema Layer
- Sample Program Listings for IRD Data Layer
- Sample Outputs of IRD Reports
- Sample Screen Programs for IRD Layers

Note: Only the first two items of the above are attached to this
Appendix.

List of IRD Schema Layer Programs

A list of report and screen programs for the IRD Schema layer is given below:

<u>Reports Name</u>	<u>Description</u>
ENT_REL_REP.SQL	Entity Relationship Report
ENTITY_REP.SQL	Entity Types Report
ATT_REP.SQL	Attribute Types Report
REL_CL_REP.SQL	Relationship Class Types Report
REL_REP.SQL	Relationship Report
<u>Query Screens</u>	<u>Description</u>
Q_ENT_TYPE.INP	Query/report screen for Entity Types
Q_REL_CL_TYPE.INP	Query/report screen for Relationship Class Types.
Q_ATT_TYPE.INP	Query/report screen for Attribute Types
REL_TYP.INP	Query/report screen for Relationship Types
Q_REL_ATT.INP	Relationship Attribute

<u>Maintenance Screens</u>	<u>Description</u>
ENT_TYPE.INP	Maintenance screen for Entity Types
REL_CL_TYPE.INP	Maintenance screen for Relationship Class Types
ATT_TYPE.INP	Maintenance screen for Attribute Types report
ENT_REL.INP	Entity Types linked by Relationship Class Types
ENT_TYP_CONT_ATT_TYP.INP	Query/report screen for Entity Attribute
REL_TYP_CONT_ATT_TYP.INP	Query/report screen for Relationship Attribute
Q_ENT_ATT.INP	Query/report screen for Entity Attribute
Q_REL_ATT.INP	Relationship Attribute

List of IRD Data Layer Programs

A list of report and screen programs for the IRD Data Layer is given below:

<u>Reports</u>	<u>Description</u>
DATABASE_REP.SQL	Database Entities Report
ELE_REP.SQL	Element Entities Report
<u>Screens</u>	<u>Description</u>
Q_DB_ENT.INP	Query/Report screen for Database Entities
Q_SYS_ENT.INP	Query/Report screen for System Entities
<u>Maintenance Screens</u>	
APPL.INP	Maintenance Screen for System entities
DATABASE.INP	Maintenance Screen for Database entities

Appendix E6 - MDAS Administration Programs

- **Tables for MDAS Administration**
- **List of Programs for MDAS Administration**
- **Sample Menu and Screen Display Printouts**
- **Screen Program Listing**

Note: Only the first two items of the above are attached to this
Appendix.

Table for MDAS Administration

The schema descriptions for the tables associated with MDAS

Administration are given below:

<u>Table</u>	<u>Column Names</u>	<u>Descriptions</u>
MDAS__USER_ROLE	ROLE_ID ROLE_DESC	Role in MDAS Role description
MDAS_USER_INFO	USER_ID ROLE_ASSIGNED DATE_ENROLLED ENROLLED_BY	User's database ID Role in the Toolset Enrollment date Enrolled by admin.

List of Programs for MDAS Administration

A list of reports and screens for the MDAS Administration program is given below:

<u>Application Admin. Program</u>	<u>Program Descriptions</u>
USER_ROLE.INP	Maintenance screen for MDAS user roles
MDAS_USER_INFO.INP	Screen for MDAS user maintenance
GRANT_ADMIN.SQL	Grant script for MDAS Administrator
GRANT_DBA.SQL	Grant script for DBA users

END

24-1 2-92

FIN