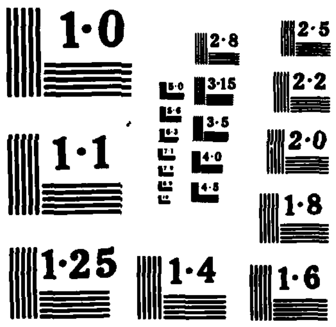
AD-A154 095 DESIGN AND IMPLEMENTATION OF AN INTELLIGENCE DATABASE (U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA J E JANG DEC 84 1/1

UNCLASSIFIED F/G 9/2 NL


1·0
2·8
2·5
3·15
2·2
3·5
1·1
4·0
2·0
4·5
1·8
1·25
1·4
1·6

NAVAL POSTGRADUATE SCHOOL

Monterey, California

DEPARTMENT OF THE NAVY · NAVAL POSTGRADUATE SCHOOL

THESIS

DESIGN AND IMPLEMENTATION OF
AN INTELLIGENCE DATABASE

by

JANG, Jai Eun
December 1984

Thesis Advisor: Samuel H. Parry

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A154095	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Design and Implementation of an Intelligence Database		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis December 1984
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) JANG, Jai Eun		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943		12. REPORT DATE December 1984
		13. NUMBER OF PAGES 103
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Semantic Database Model, Functional Dependency, ORACLE, DBTG, Schema, Normal Form

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis presents che design and implementation of the Intelligence Database system. A database management system must be used in Intelligence System in order to increase end-user productivity, decrease staff effort, enable the work to be done more efficiently, and permit end-user management more authority and responsibility. The Semantic Database Model was chosen as the method for designing the database. (Continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The SDM is a high-level semantics-based database description and structuring formalism for database design and enhances usability of database system. Using the output of SDM in the Intelligence database, the records are rearranged in order to fit a relational DBMS. The Intelligence database is implemented, using the ORACLE relational DBMS.

Design and Implementation of an Intelligence Database

by

JANG, Jai Eun
Captain (P), The Republic of Korea Army
B.A., Korea Military Academy, 1978

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1984

Author: Jang, Jai Eun
JANG, Jai Eun

Approved by: [illegible]
Samuel H.Parry, Thesis Advisor

[illegible]
David H. Hsiao, Second Reader

[illegible]
Bruce J. MacLennan, Chairman
Department of Computer Science

[illegible]
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

This thesis presents the design and implementation of the Intelligence Database system. A database management system must be used in Intelligence System in order to increase end-user productivity, decrease staff effort, enable the work to be done more efficiently, and permit end-user management more authority and responsibility. The Semantic Database Model was chosen as the method for designing the database. The SDM is a high-level semantics-based database description and structuring formalism for database design and enhances usability of database system. Using the output of SDM in the Intelligence database, the records are rearranged in order to fit a relational DBMS. The Intelligence database is implemented, using the ORACLE relational DBMS. Additional keywords: computer architecture; systems engineering; relational data bases.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

My country, Republic of Korea, gave me a chance to study the graduate course of computer science. I am very grateful to many people for their help. It is time for me to work harder than before for my lovely country.

Most of all, I would like to express my appreciation to my thesis advisor, Professor Samuel Parry, for his enthusiastic guidance and support. Without the many hours of counseling and endless supply of ideas he provided, this thesis could not have been completed.

I am very thankful my mother, my family, wife Yeon-Ook, daughter Ja-Yun, son Gun-Hong, for their understanding and encouragement, and for the sacrifice they have made during my study. "Hi, Ja-Yun, from now on, I have much free time to play with you".

Finally, when I return my country, I want to implement this type of Intelligence Database System in order to enhance the combat readiness of the Republic of KOREA ARMY against the North Korea threat. This will be accomplished using Semantic Database Model and ORACLE relational database management system.

I. INTRODUCTION

It is obvious that it is the database system era in computer technology and applications. Database processing has grown significantly in computer science areas and also in management of certain organizations.

An important consideration in database development is to store data in such a way that it can be used for a wide variety of applications and can be changed and quickly and easily. To achieve the flexibility of data usage, three aspects of database design and implementation are important. First, the data should be independent of each other and functionally dependent on the key value. Second, it should be possible to interrogate for user's requirements using application programs or the DBMS itself. Third, these data items should provide useful information for decision makers to analyze, to investigate, to plan and to manage in a certain organization.

It is very difficult to develop database systems which perform in an optimal fashion. There are many different ways n which data can be structured and each has its own advantages and disadvantages. Different users want to use different data/information. It is hardly possible to satisfy all of the users with one type of data organization.

3. Evaluation A Conceptual Design

What makes a good conceptual design? It is possible to itemise a useful set of properties that characterise a good conceptual design as follows:

* Concept complete : guarantees not only that useful objects are not left out of the database but also that physical database designers are not inappropriately constrained. It is true that for many derived concepts the derivation can only be made in one direction.
* Unbiased toward applications : groupings which favor one application at the expense of others should be identified and removed when possible.
* Evolvable : it should be locally modifiable and it should be flexible in supporting user interpretations.
* Independence of existing installation and DBMS constraints : initially tailoring a design to fit the limitation of the current state of its intended support system makes it difficult to separate out these restrictions when the support system changes or is replaced. The better approach is to develop the design independence of such limitations and conventions first, then tailor it to the system.

4. Design Tools And Methodologies

The primary tool in database design is the language used to specify the design. Such a specification language is a tool in the sense that its vocatulary and

* The structure of the database's conceptual objects
* The structure of its basic functions and update procedures
* Integrity constraints on the database.

The conceptual objects of a database are all very important to the running of an enterprise whether they be people, procedures, events or the inter-relationships among these. Such objects must be grouped into types which identify their significant attributes and processing constraints.

Because a major goal of database management is data sharing, it is expected that the updates of each user will be apparent to the other users of the data. This makes it important that the necessary side effects of such changes be understood and correctly implemented by all application groups. This can be facilitated by including in the conceptual design specification of the basic update operations for objects in the database.

It is also useful for the onceptual design to include, via function and procedure specification, conventions for naming individuals that exhibit a correct sensitivity to updates. In addition to the integrity constraints maintained by the primitive update operations and those enforced by the type declarations, there may be many more sophisticated constraints that must be maintained for the database.

The next phase which may be called 'conceptual design' is the integration of all the concepts which are necessary to support the various application views. In effect, conceptual design is the production of a 'community' model in which the idiosyncrasies of the individual views are resolved. At the conceptual level, data should appear in a structure which is most perspicuous for concept integration. It should explicitly define how concepts are related one to another; it should not contain any implementation detail; and it should be locally modifiable.

The final phase, 'physical design', is the mapping of the conceptual model on to physical computing devices. In this phase, performance considerations must be analyzed and shown compatible with application requirements. With most database management systems, the physical mapping is partially hidden and 'tuning' is allowed on only a fixed set of parameters.

2. The Contents Of A Conceptual Design

The conceptual design of a database serves two functions. It is used in interactions with applications programmers to verify the correctness of the program being developed. It is also used as a guideline for the physical designers - specifying to them what must be implemented without constraining how it is implemented. To achieve these objectives the following kinds of information must be determined in the design process:

1. The Level Of Database Design

A database must encompass all aspects of the data to be stored - beginning with details of how it is presented to different users and ending with how it is to be represented on the hardware of a particular installation. To achieve this in an orderly and correct fashion, the design process has been structured into the three distinct phases shown in Figure 3.1. The first phase, which may be called 'view design', is the identification and design of interfaces for the different end-user groups. Each end user requires a particular 'view' of the database to support his own application idiosyncrasies. A view should present data in the structure which is most effective for the user. This may be reports, natural language text. The view must provide tailored update facilities to manipulate the database.

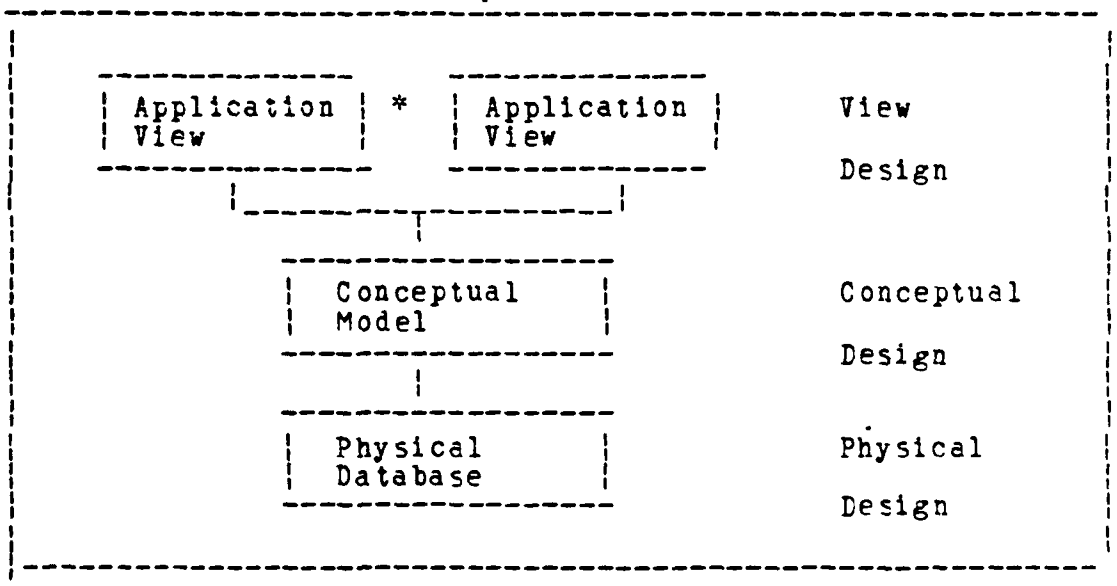


Figure 3.1 Phases in Database Design

III. INTRODUCTION TO DATABASE DESIGN

A. CONCEPTUAL DATABASE DESIGN

Database Management Systems have evolved from file systems to answer two critical needs: support for more inter-related data and support for sharing data among many diverse applications. These goals are being achieved, in part, by providing DBMS software to physically link related data into complex structures using such mechanisms as pointer chains, indices and sequential positioning. They are also achieved by the development of database design methodologies and rules.

To reduce the complexity of using DBMSs, designers have developed special interfaces to these systems that decompose their use into easily understood phases. Thus, most DBMSs have Data Description Languages (DDLs), Data Manipulation Languages (DMLs) and Query Languages. The DDL is used to specify the design of the database. The DML is used to generate application programs that access the database in terms of the objects specified using the DDL. The Query Language is used for more ´casual´ database accesses. The DML is orinted toward the development of database access programs that are efficient to execute while Query Languages are orinted towards ease in writing such programs.

the same data so that applications can use data in a format that is familiar and useful to them.

The DBMS also has features to provide security over data; the tools provided ensure that only authorized data are accessed. Also, the DBMS controls concurrent processing and includes features to provide backup and recover.

The final type of program involved in database processing is the operating system. This set of programs controls the computer's resources. The DBMS sends requests for input/output services to operating system.

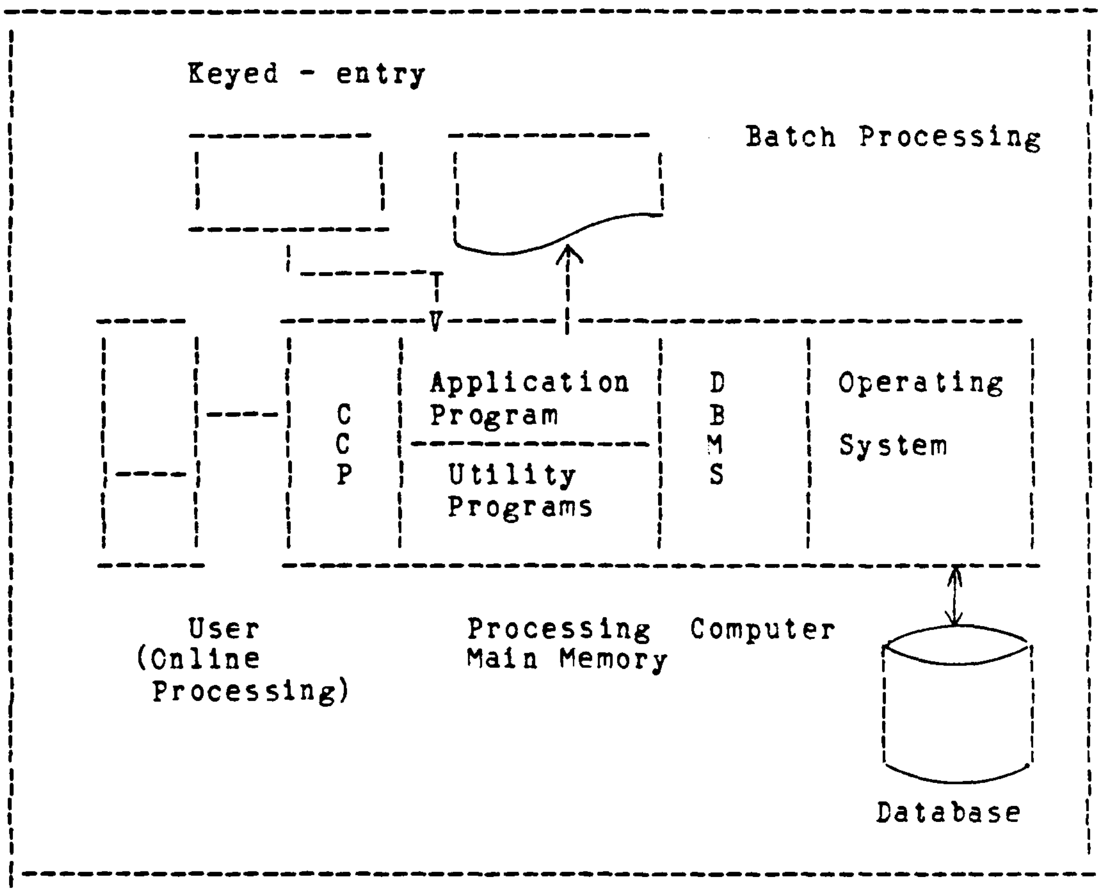


Figure 2.2 Programs Involved in Typical Database Processing

DBMS intercepts the request and interprets it; (3) the DBMS inspects, in turn, the extenal schema, the external/conceptual mapping, the conceptual schema, the conceptual/internal mapping, and the storage structure definition; and (4) the DBMS performs the necessary operations on the stored database.

D. PROGRAMS IN TYPICAL DATABASE PROCESSING

Figure 2.2 shows the approximate relationships of the major types. Online processing requests or transactions are provided by users at terminals. The requests are sent to the processing computer over communications lines.

The communications control program (CCP) has several important functions. It provides comunications error checking and correction, coordinates terminal activity, routes messages to the correct next destination, and formats messages for various types of terminal equipment.

The utility programs are provided by either the DBMS or the hardware vendor. These programs provide a wide variety of services. Query/update utilities provide generalized retrieval and update of the database.

For normal processing, the DBMS receives data and stores it for subsequent processing. This system acts as a sophisticated data librarian. The DBMS allows application programs and utilities a wide variety of access strategies. It also enables these programs to have different views of sequence, indexing, hash-addressing, or any other storage/ access details. The conceptual model is a view of the total database content, and the conceptual schema is a definition of this view. The definition in the conceptual schema is intended to include a great many additional features, such as the authorization checks and validation procedures.

The internal model is a very low-level representation of the entire database; it consists of multiple occurrences of multiple types of internal records. The internal model is described by means of the internal schema, which not only defines the various types of stored records but also specifies what indexes exist, how stored fields are represented, what physical sequence the stored records are in, etc.

The conceptual/internal mapping defines the correspondence between the data model and the stored database; it specifies how conceptual records and fields map into their stored counterparts. If the structure of the stored database is changed - if a change is made to the storage structure definition - the conceptual/internal mapping must be changed accordingly, so that the conceptual schema may remain invariant.

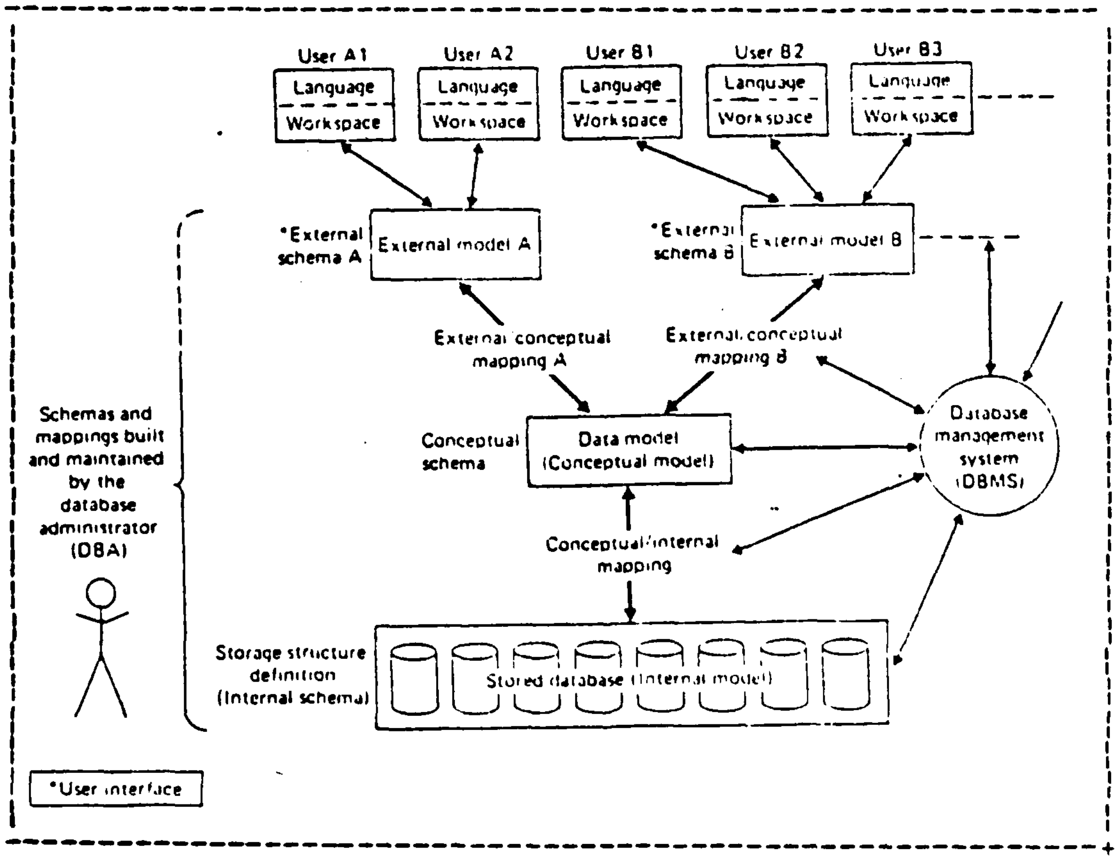
The Database Management System is the software that handles all access to the database. Conceptually what happens is the following : (1) A user issues an access request, using some particular data sublanguage; (2) the

Next, the various components of the system will be examined. The users are either application programmers or remote terminal users of any degree of sophistication. Each user has a language at his disposal. It will be a conventional programming language, such as COBOL, PL/1, etc.

Each user is provided with a workspace, which acts as the receving or transmitting area for all data tranferred between the user and the database. The user is said to view the database by means of an external model. An external model is thus the information content of the database as it is viewed by some particular user.

Each external model is defined by means of an external schema, which consists of descriptions of each of the various types of external records in that external model. In addition, there must be a definition of the mapping between the external schema and the undering conceptual schema.

The conceptual model is a representation of the entire information content of the database, again in a form that is somewhat abstract in comparison with the way in which the data is physically stored. The conceptual model is defined by means of the conceptual schema, which includes definitions of each of the various types of conceptual records. If data independence is to be achieved, these definitions must not involve any considerations of storage structure or access strategy. Thus there must be no reference to stored field representations, physical proposed by the ANSI/SPARC Study Group on Data Base Management Systems. The architecture is divided into three general levels: internal, conceptual, and external. Broadly speaking, the internal is the one closest to physical storage, the one concerned with the way in which the data is actually stored; the external level is the one closest to the users, that is, the one concerned with the way in which the data is viewed by individual users; and the conceptual level is a 'level of indirection' between the other two.



User A1
User A2
User B1
User B2
User B3
Language
Workspace
*External schema A
External model A
*External schema B
External model B
External/conceptual mapping A
External/conceptual mapping B
Conceptual schema
Data model (Conceptual model)
Database management system (DBMS)
Conceptual/internal mapping
Storage structure definition (Internal schema)
Stored database (Internal model)
Schemas and mappings built and maintained by the database administrator (DBA)
*User interface

Figure 2.1 An Architecture for a Database System

implemented, operating costs for some systems will be higher. Sequential processing, for example, will never be done as fast as in the database environment, because of the extra overhead.

Another disadvantage is that database processing tends to be complex. Large amounts of ata in many different formats can be interrelated in the database. Both the database system and the application programs must be able to process these structures. This requires more sophisticated programming. Backup and recovery are more difficult in the database environment. This is because of increased complexity and because the database is often processed by several users concurrently. Determining the exact state of the database at the time of failure may be a problem. Given that, it may be even more difficult to determine what should be done next.

The third disadvantage is that integration, and hence centralization, increases vulnerability. A failure in one component of an integrated system can stop the entire system. This event is especially critical if, as is often the case, the operation of the user organization depends on the database.

C. AN ARCHITECTURE FOR A DATABASE SYSTEM

The architecture of a database is outlined in Figure 2.1 **[Ref. 4]**. This figure is in broad agreement with that

1. Advantage of Database

First, database processing enables more information to be produced from a given amount of data. Second, the elimination or reduction of data duplication saves file space, and to some extent, can reduce processing requirements. The most serious problem of data duplication is that it can lead to a lack of data integrity. A common result of a lack of data integrity is conflicting reports. Third, creation of program/data independence - the immunity of applications to change in storage structure and access strategy, which implies that the application concerned do not depend on any one particular storage structure and access strategy. Another advantage is better data management. When data is centralizied in a database, one department specializes in the maintenance of data. That department can specify data standards and ensure that all data adhere to the standards. When someone has a data requirement, he can contact one department instead of many file maintenance groups. F rthermore, centralization of data management leads to economies of scale.

2. Disadvantage of Database

A major disadvantage of database is that it can be expensive. The DBMS may occupy so much main memory that additional memory must be purchased. Even with more memory, it may monopolize the CPU, thus forcing the user to upgrade to a more powerful computer. Once the database is

II. BASIC CONCEPT OF DATABASE

A. WHAT IS A DATABASE?

First of all, there is the database itself - a collection of data stored on disks, drums or other secondary storage media. Second, there is a set of ordinary batch application programs which run against this data, operating on it in all the usual ways. Third, the database is 'integrated'. This means that the data base contains the data for many users, not just for one, which in turn implies that any one user will be concerned with just a small portion of it. According to [Ref. 4], the definition of database is a collection of stored operational data used by the application systems of some particular enterprise. Some examples of enterprise are manufacturing companies, banks, hospitals, etc.

B. WHY DATABASE?

There are many answers to this question. One general answer is that it provides the enterprise with centralized control of its operational data. This is in sharp contrast to the situation that prevails in most enterprises today, where typically each application has its own private files so that the operational data is widely dispersed, and there is little or no attempt to control it in a systematic way.

The normal form concepts of relational database will be used to develop an Intelligence Database, because the Relational Database Management System supports independence better than other models and is easier to implement.

Chapter II addresses the basic concepts of database, which relates to the database system development for the Intelligence Database. Chapter III addresses the introduction to database design, which includes conceptual database design and physical database design. Chapter IV describes how the Intelligence Database is designed using Semantic Database Model. First of all, the SDM is designed; then a relational or network model is applied and implemented. Chapter V describes Relational database design, which includes relational Normal Forms and the characteristics of relational database and conversion of SDM into Relational database design. Chapter VI addresses the implementation which is implemented on the ORACLE Database Management System. Finally, Chapter VII presents conclusion and recommendations based on the research presented in the thesis.

syntax shapes the way designers percieve the application they are modelling. A model too primitive in its vocabulary requiries more complicated concepts to be built up, producing a specification that is difficult to understand and therefore to use and to verify.

Each of the following properties contribute to value of a good data model :

1. It should be expressive : a data model that is sensitive to important distinctions will guide its users to include the concepts and objects necessary to a good design.
2. It should not overconstrain implementors : because a conceptual design is the mechanism used to instruct physical database implementors the model on which it is based should not imply particular implementation strategies.
3. A data model should have a formal basis: this relieves the designer of ambiguity and provides the physical designers and implementors with a sound foundation for verifying their work.
4. A data model should be widely applicable: A conceptual design for an extensive enterprise may need to encompass applications that are very dynamic in terms of interactions among the different objects of interest
5. A data model should be understandable : A conceptual design for an extensive enterprise can be both very

large and very complex. To show even a part of a specification to an end user to check its correctness, it is necessary that the data model in which it is expressed provides some kind of non-technical presentation mode.

5. Implementation Design Components

A diagram of the spectrum of inputs to and outputs from the implementation design is shown in Figure 3.2.

```
-------------------------------------------------------------------
|                                                                 |
|                ------ Conceptual design -------                 |
|                |               |               |                |
|                |               |               |                |
|                V               V               V                |
|             Volume            DBMS         HIgh - level         |
|            and usage      independent       program            |
|          qualification      schema          specs               |
|                 |               |             |                  |
|                 |               |             |                  |
|                 V               V             V                  |
|               ----------------------------------                 |
|     DBMS      |                                |                 |
| characteristics ->|  Implementation  Design    |                 |
|               |                                |                 |
|               ----------------------------------                 |
|                 |           |            |        |             |
|                 |           |            |        |             |
|                 |           V            |        V             |
|                 |          DBMS          |     Program          |
|                 |       processible      |      design          |
|                 |         schema         |     guidance         |
|                 |           |            V        |             |
|                 |           |-----> Subschema <--               |
|                 V           V                                   |
|            Guidance to  Specification for                       |
|             database       physical                             |
|            operations       design                               |
|              group                                              |
|                                                                 |
|-----------------------------------------------------------------
```

Figure. 3.2 Implementation design environment.

Inputs are as follows;

1. DBMS-independent schema - The major result of the conceptual design phase, to be refined by the implementation design phase.
2. Operational requirements quantification - Specification for integrity, recovery, security, and response time limits.
3. Volume and usage quantification - Database size in terms of data occurences and application frequencies.
4. Consistency constraints - Rules for keeping data elements consistent, rules for dealing with inconsistent data.

Outputs are as follows;

1. DBMS-processible schema - Specifications for a database structure that can be implemented with a specific DBMS.
2. Subschemas - DBMS-processible database structure consistent with individual user views and security constraints.
3. Guidance to the database operations group - a summary of requirements, constraints, and available data on the hardware/software environment to the DBA.

B. PHYSICAL DATABASE DESIGN

The second stage of database design -physical design- is a stage of transformation. The logical schema is tranformed into the particular data constructs that are available with the DBMS to be used. Whereas the logical design is DBMS-independent, the physical design is very much DBMS-dependent. Detailed specifications of the database structure are produced. These specifications will be used during implementation to write source statements that define the database structure to the DBMS. These statements will be compiled by the DBMS and the object form of the database structure will be stored within the database. as illustrated in Figure 3.3. [Ref. 3]

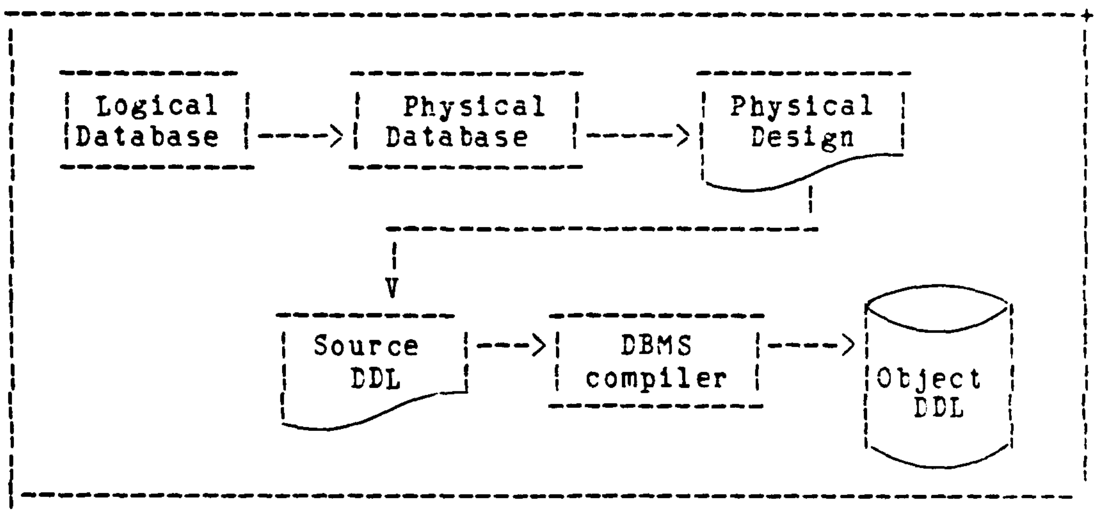


Figure 3.3 Role of Physical Design

1. Physical Design Environment

The design environment is basically the same for both file design and physical database design. However, many design decisions for files are much simpler than for multiple-record-type design. First, the major categories of inputs and outputs for the physical design phase are illustrated in Figure 3.4.

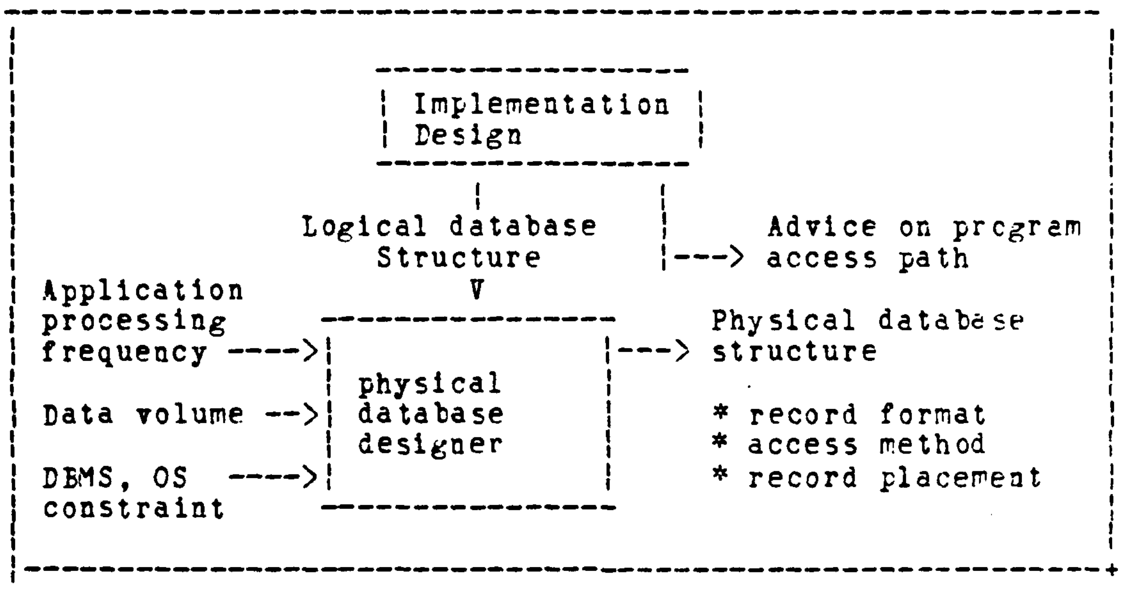

Implementation Design
Logical database Structure
Advice on program access path
Application processing frequency
Data volume
DBMS, OS constraint
physical database designer
Physical database structure
* record format
* access method
* record placement

Figure 3.4 Physical Environment

In general, physical design considers new parameters, but previous tentative decisions on access paths and record allocation are finalized in this phase. Parameters regarding data volume, application processing frequency, and sequence of operations in aplication programs are the same as those required for implementation design. New parameters introduced at this stage are those specific to DBMS and operating system access methods, those specific to describe physical device capacity limitations and timing characteristics and all operational requirements.

The visible components of the resulting physical database structure are the stored record format, stored record placement specification, and access method specification. Underlying these specifications is the satisfaction of all operational requirements and hardware/software system constraints. During the design process, consideration of efficiency issues can take place only after the various constraints are satisfied and a feasible solution has been obtained.

2. Performance Measure

The determination of performance measures for physical design is most critical to the design process. It affects not only the design choices, but also the techniques employed to determine those choices.

Let us assume that database system performance will be described in terms of cost. At various times cost may be given in terms of time, space, or possibly monetary value. Returning to our discussion of the database system life cycle, we can describe the total cost of the life cycle in terms of the following:

* Planning cost
* Design cost : programs, database
* Implementation and testing cost : programs, databases
* Operational costs : users, compute resource
* Maintenance costs : program errors, data integrity loss

3. Outputs of Physical Design

In general, two major specifications are produced. First, the physical specification of the logical schema is defined. It is the physical schema. This schema is a transformation of the logical schema into the data modeling constructs available with the DBMS to be used. Second, user views are defined.

a. Physical schema

The contents of records must be defined, and the name and format of each field of each record specified. Constraints from the logical database design are tranformed into critiria for field descriptions. Keys of database records need to be identified, and overhead structures for supporting the keys defined. Record relationships are also defined in the physical design.

b. User views

User views are generally a subset of the schema. Records or relationships may be omitted from a view; fields may be omitted or rearranged. Also, the names of records, fields, or relationships may be changed. This flexibility allows users to employ terminology that is familiar and useful to them.

C. APPLICATION OF DATABASE MODELS TO DATABASE DESIGN

Figure 3.5 shows the major steps involved in designing a database. Inputs to design are statements of data requirements from the specification data directory. The output of design is a specification that can be used to implement the database using a commercial DBMS. The design that is produced depends very much on the DBMS to be employed. For this reason, Figure 3.5 shows two alternative design outputs. If we are going to use a DBMS based on the relational model, we will produce a relational design. If we are going to use a DBMS based on the CODASYL DBTG model, we will produce a DBTG(network) design.

Within this figure are two steps : logical(DBMS - independent) design and physical(DBMS - dependent) design. After logical design, there is a branch, depending on the DBMS to be employed. If we are going to use a relational DBMS, then the output of physical design will be a relational design expressed as relation definitions and supporting documentation.

If we are going to use a CODASYL DBMS, then the output of the physical design will be a CODASYL design expressed as data structure diagrams and supporting definitions.

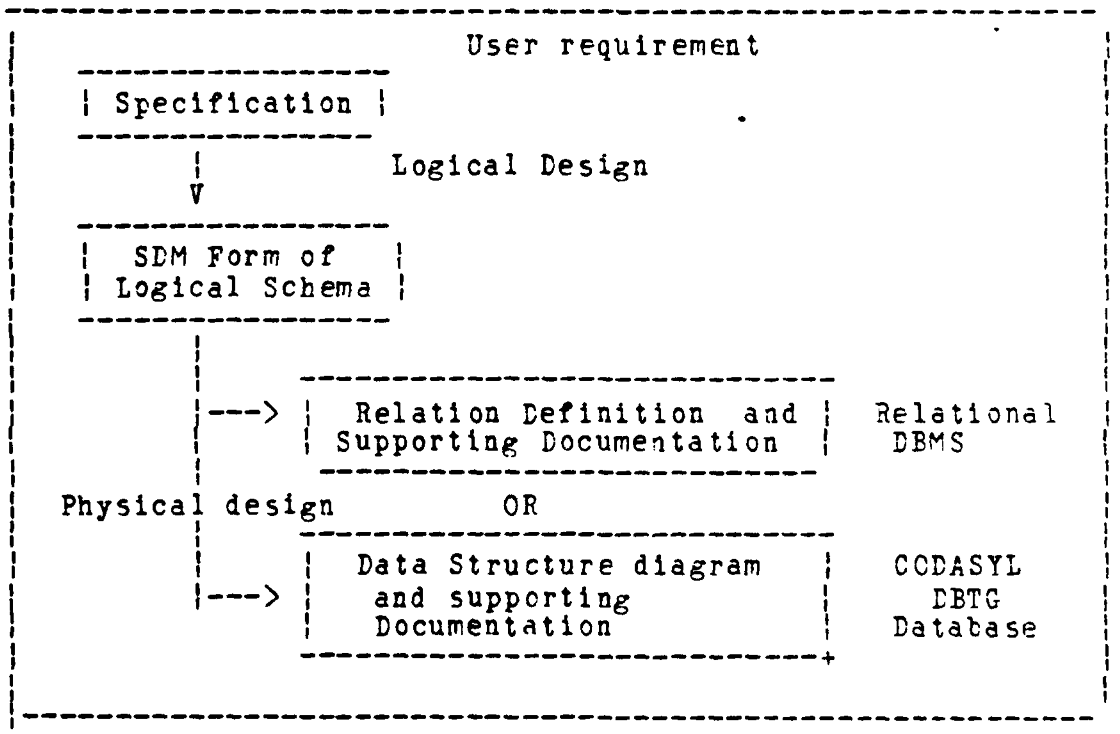

User requirement
Specification
Logical Design
SDM Form of Logical Schema
Relation Definition and Supporting Documentation
Relational DBMS
Physical design
OR
Data Structure diagram and supporting Documentation
CODASYL DBTG Database

Figure 3.5 Use of Model in Database Design

IV. SEMANTIC DATABASE MODEL

The Semantic Database Model(SDM) was developed by Hammer and McLed [Ref. 8] and first published in 1981. It will be used as the tool for expressin a logical schema for the Intelligence database design.

SDM is a high-level semantics-based database description and structuring formalism for the database. This database model is designed to capture more of the meaning of an application environment than is possible with contemporary database models.

SDM is designed to enhance the effectiveness and usability of database systems. An SDM database description can serve as a formal specification and documentation tool for a database. It can provide a basis for supporting a variety of powerful user interface facilities, serve as a conceptual database model in the database design process, and be used as the database model for a new kind of database management system.

A. INTRODUCTION

Every database is a model of some real world system. At all times, the contents of a database are intended to represent a snapshot of the state of an application environment, and each change to the database should reflect an event occuring in that environment. Therefore, it is appropriate that the structure of a database mirror the structure of the system that it models. A database whose organization is based on naturally occurring structure will be easier for a database designer to construct and modify than one that forces him to translate the primitives of his problem domain into artificial specification constructs.

The global user view of a database, as specified by the database designer, is known as its logical schema. A schema is specified in terms of a database description and structuring formalism and associated operations, called a database model. It was thought that the data structures provided by contemporary database models do not adequately support the design, evolution, and use of a complex database. These database models have significantly limited capabilities for expressing the meaning of a database and relating a database to its corresponding application environment. The semantics of a database defined in terms of these mechanisms are not readily apparent from the schema; instead, the semantics must be separately specified by the database designer and consciously applied by the user.

The goal is the design of a higher-level database model that will enable the database designer to naturally and directly incorporate more of the semantics of a database into its schema. Such a semantics-based database description and structuring formalism is intended to serve as a natural application modeling mechanism to capture and express the structure of the application environment in the structure of the database.

1. The Design of SDM

In designing SDM, many database aplications were analyzed in order to determine the structures that occur and recur in them. The shortcomings of contemporary database models in capturing the semantics of these applications were assessed, and the strategies were developed to address the problems discovered. This design process was iterative, in that features were removed, added, and modified during various stages of esign.

SDM has been designed with a number of specific types of uses in mind. First, SDM is meant to serve as a formal specification mechanism for describing the meaning of a database: SDM provides a precise documentation and communication medium for database users. In particular, a new user of a large and complex database should find its SDM schema of use in determining what information is contained in the database. Second, SDM provides the basis for a variety of high-level semantics-based user interfaces to a database; these interface facilities can be constructed as front-ends to existing database management systems.

SDM has been designed to satisfy a number of criteria that are not met by contemporary database models, but which I believe to be essential in an effective database description and structuring formalism. They are as follows.

The constructs of the database model should provide for the explicit specification of a large portion of the meaning of a database. Many contemporary database models (such as the CODASYL DBTG network model and the hierachical model) exhibit compromises between the desire to provide a user-oriented database organization and the need to support efficient database storage and manipulation facilities. In contrast, the relational database model stresses the separartion of user-level database specification and underlying implementation detail.

However, the Semantic expressiveness of the hierachical, network, and relational model is limited; they do not provide sufficient mechanism to allow a database schema to describe the meaning of a database. Such models employ overly simple data structures to model an application environment. In so doing, they lose information about the database; they provide for the expression of only a limited range of a designer's knowledge of the application environment. It is necessary to break with the tradition of record-based modeling and to base a database model on structual constructs that are highly user oriented and expressive of the application environment.

A database model must support a relativist view of the meaning of a database, and allow the structure of a database to support alternative ways of looking at the same information. In order to accommodate multiple views of the same data and to enable the evolution of new perspectives on the data, a database model must support schema that are flexible, potentially logically redundant, and integrated. Flexibility is essential in order to allow for multiple and coequal views of the data.

Contemporary, record-oriented database models do not adequately support relativism. In these models, it is generally necessary to impose a single structural organization of the data, one which inevitably carries along with it a particular interpretation of the data's meaning. This meaning may not be appropriate for all users of the database and may become entirely obsolete over time.

Another consequence of the primacy of the principle of relativism is that, in general, the database model should not make rigid distinctions between such concepts as entity, association, and attribute. Higher-level database models that do require the database schema designers to sharply distinguish among these concepts are thus considered somewhat lacking in their support of relativism.

A database model must support the definition of schemata that are based on abstraction entities. Specifically, this means that a database model must facilitate the description of relevant entities in the application environment, collections of such entities, relationships among entities, and structual inter-collections among the collections.

Allowing entities to represent themselves makes it possible to directly reference an entity from a related one. In record-oriented database models, it is necessary to cross reference between related entities by means of their identifiers. While it is of course necessary to eventually represent 'abstract' entities as symbols inside a computer, the point is that users should be able to reference and manipulate abstractions as well as symbols.

B. A SPECIFICATION OF SDM

The following general principles of database organization underlie the design of SDM [Ref. 3].

(1) A database is to be viewed as a collection of entities that correspond to the actual objects in the application environment

(2) The entities of a database are organized into CLASSES that are meaningful collections of entities.

(3) The classes of a database are not in general independent, but rather are logically related by means of Interclass connections.

(4) Database entities and classes have ATTIRIBUTES that describe their characteristics and relate them to other database entities. An attribute value may be derived from other values in the database.

select representations for these constructs in a routine, if not algorithmic, fashion.

SDM provides an effective base for accommodating the evolution of the content structure and use of a database. Relativism, logical redundancy, and derived infomation support this natural evolution of the schema.

A related use of SDM is as a medium for documenting a database. One of the more serious problems facing a novice user of a large database is determining the information content of the database and locating in the schema the information of use to him. An SDM schema for a database can serve as a readable description of its contents, organized in terms that a user is likely to be able to comprehend and identify.

schema using the database model of the DBMS to be employed, is a difficult and error-prone procedure. A primary reason for this difficulty is the gap between the semantic level of the application and the data structures of the database model; the DBA must bridge this gap in a single step, simultaneously conducting an information requirements analysis and expressing the results of his analysis in terms of the database model.

1. The Advantage Of SDM

An SDM schema will serve as a specification of the information that the database will contain. All too often, only the most vague and amorphous English language descriptions of a database exist prior to the database design process. A formal specification can more accurately, completely, and consistently communicate to the actual designer the prescribed contents of the database. SDM provides some structure for the logical database design process. The DBA can first seek to describe the database in high-level semantic terms, and then reduce that schema to a more conventional logical design.

SDM supports a basic methodology that can guide the DBA in the design process by providing him with a set of natural design templates. That is, the DBA can approach the application in question with the intent of identifying its classes, subclasses, and so on. Having done so, he can available for describing attributes that d not match any of these cases. For example, Total-foe is derived from IREC record by calculating total number of foes.

2. Class Attribute Interrelationships

Attribute derivation primitives analogous to primitives for member attributes can be used to define derived class attributes, as these primitives derive attribute values from those of other attributes. In addition, there are two other primitives that can be used in the definition of derived class attributes.

(1) An attribute can be defined so that its value equals the number of members in the class it modifies. For example, Total-foe has the derivation 'number of members in this class'.

(2) an attribute can be defined whose value is a function of a numeric member attribute of a class; the functions supported are 'maximum', 'minimum', 'average'.

D. APPLICATION

SDM is simply an abstract database modeling mechanism and language that is not dependent on any supporting computer system. One set of applications uses SDM in precisely this mode to support the process of defining and designing a database as well as in facilitating its subsequent evolution. It is well known that the process of logical database design, wherein the DBA must construct a

If A1 is a multivalued attribute, then it is permissible for each member of C1 to match to several members of C2; in this case, the collection of A3 values is the value of attribute A1. For example, Iclass/Itype is matched with Pclass/Ptype (Figure 4.3).

Therefore, match is defined only in IREC record and not defined in PREC record. That means, according to PID, Pclass/Ptype is matched and the value is updated.

Inversion and matching provide multiple ways of viewing n-ary associations among entities. Inversion permits the specification of binary associations, while matching is capable of supporting binary and higher degree associations.

c. Derivation

Inversion and matching are mechanisms for eastablishing the equivalence of different ways of viewing the same essential relationships among entities. SDM also provides the ability to define an attribute whose value is calculated from other information in the database. Such an attribute is called Derived, and the specification of its computation is its associated derivation.

The approach is to provide a small vocabulary of high-level attribute derivation primitives that directly model the most common types of derived information. Each of these primitives provides a way of specifying one method of computing a derived attribute. More general facilities are attribute A1 of class C1 can be specified as the inverse of member attribute A2 of C2 which means that the value of A1 for a member M1 of C1 consists of those members of C2 whose value of A2 is M1. The inversion interattribute relationship is specified symmetrically in that both an attribute and its inverse contain a description of the inversion relationship. A pair of inverse attributes in effect establish a binary association between the members of the classes that the attributes modify. For example, Wammo in WREC record has inverse relationship with Where-needed in AREC record.

Therefore, value class and inverse is defined in Wammo and another item name is defined in AREC record, which corresponds to Wammo item name.

b. Matching

The second way in which a member attribute can be related to other information in the database is by matching the value of the attribute with some member of a specified class. In particular, the value of the match attribute A1 for the member M1 of class C1 is determined as follows.

(1) A member M2 of some class C2 is found that has M1 as its value of member attribute A2.

(2) The value of member attribute A3 for M2 is used as the value of A1 for M1.

(4) An (optional) ATTRIBUTE DESCRIPTION is text that describes the meaning and purpose of the attribute.

(5) The attribute is specified as either SINGLE VALUED or MULTIVALUED. The value of a single-valued attribute is a member of the value class of the attribute, while the value of a multivalued attribute is a subclass of the value. (e.g., Pclass/type has Multi-value)

(6) An attribute can be specified as MANDATORY, which means that a null value is not allowed for it.(e.g., Iclass)

(7) An attribute can be specified as not changeable, which means that once set to a nonnull value, this value cannot be altered except to correct an error. (e.g., Iclass)

(8) A member attribute can be required to be EXHAUSTIVE of its value class. This means that every member of the value class of the attribute must be the value of some entity.

(9) A multivalued member attribute can be specified as NONOVERLAPPING, which means that the values of the attribute for two different entities have no entities in common.

1. Member Attribute Interrelationships

a. Inversion

The first way in which a pair of member attributes can be related is by means of INVERSION. Member

D. ATTRIBUTE

In Figure 4.6 above, each class has an associated collection of attributes. Fach attribute has the following features.

(1) An attribute name identifies the attribute. An attribute must be unique with respect to the set of all attribute names used in the class, the class's underlying base class, and all *eventual* subclass *of* that the base class (e.g., Iclass, Iid) in IREC (Figure 4.2).

(2) The attribute has a value which is either an entity in the database or a collection of such entities. The value of an attribute is selected from its underlying value class, which contains the permissible values of the attribute. The value of an attribute may also be the special value NULL. (e.g., INS-CLASS, INS-ID) in IREC (Figure 4.2).

(3) The APPLICABILITY of the attribute is specified by indicating that the attribute is either:

    (a) a member attribute. which applies to each member of the class, and so has a value for each member (e.g., Iclass of IREC) (Figure 4.2)

    (b) a class attribute, which applies to a class as a whole, and has only one value for the class (e.g.,Idate of IREC)

WEAPON-CLASS : subclass of STRINGS where value is AC, SH, ARU

WEAPON-TYPE : subclass of SRTIN S where value is positive single digit integer 1,2,3,4,5,6

RANGE : subclass of STRINGS where value is positive integer less than 100,000

FUEL-CAPACITY : subclass of STRINGS where value is positive integer less than 20,000

MAX-LOAD : subclass of STRINGS where value is positive integer less than 500,000

DATE : Subclass of STRINGS where value is positive integer between 1...365

INS-CLASS : subclass of STRINGS where format is 2 characters: AF, PO, AR

INS-ID : format is 3 digit positive integer

AREA : value is in between 1...100

N0-OF-PERSON : value is less than 10,000

FRIEND-OR-FOE : formats are FRD, FOE

NUM-OF-WEAPON : format is positive integer

WEATHER : value is FAIR, CLDY or PCLDY

AMMO-CAT : value is single letter

RRANGE : value is positive integer

WARHEAD : value is positive integer 1...10

Figure 4.6 Domain of Attribute

```
WREC

     description :  all  information  related  to

          weapon  class/type  and   their physical

          characteristics.

     Member attribute :

          Wclass
               Value class : WEAPON-CLASS
               Mandatory

          Wtype
               description : Weapon Type
               Value class : WEAPON-TYPE
               Mandatory

          Wff
               description  : whether  the  weapon
                               is  Friend  or  Foe
               Value class : WREC

          Wammo
               description : What sort of Ammo
                     can be  available for particular
                     type  of   weapons
               Value class : AREC
               Inverse : Wclass/Type-Needed
                    Multivalued

          Wrange
               description : Weapon range
               Value class : RANGE

          Wfuel
               description  : Fuel  capacity
                                of  weapon
               Value class : FUEL-CAP

          Wlbs
               description : Maximum Load
               Value class : MAX-LOAD
               not changeable

     identifier : Wclass + Wtype
```

Figure 4.5 SDM of WREC in the Intelligence Database

```
AREC

     description  :  Ammo  categories  and  their

                        physical   characterics

     Member attribute :

          Acat

               description : Ammo category
               Value class : AMMO-CATEGORY
               Mandatory
               Not changeable

          Where-needed

               description  :   What  kind  of
                         Weapon class/type needed
                         for this Ammo category

          Albs

               description : Weight of 1 round  of
                             Ammo

               Value class : MAX-load
               Mandatory
               Not changeable

          Akill

               description : Killing   radius
                              of Ammo

               Value class : RANGE

          Awar

               description : Type  of  warhead
                             of Ammo

               Value class : WARHEAD-CAT

     Identifier : Acat
```

Figure 4.4 SDM of AREC in the Intelligence Database

```
PREC

     description : information  about  the

        reconnaissance  date,  weapons  observed and

        weather  condition

     Member attributes :

          Pday

               description  : the  day  of  the  year
                    on which the photo taken
               Value class : DATE

          Pid

               description : Installation  Id  code
               Value class : INS-ID
               Mandatory

          Pclass/type

               dscription  :  Concatenation   of
                     weapon  class  and  type
               Value class : Weapon-class /
                             weapon-type

               Multivalued

          Pnum

               description : observed weapons
               Value class : NUM-OF-WEP

          Pwc

               description : Weather condition
               Value class : WEATHER
               Mandatory

     Idenfitier : Pday + Pid + Pclass/type
```

Figure 4.3 SDM of PREC in the Intelligence Database

```
INSTALLATION

     description   : the  basic master  file  for
       installation  representing  all  informations
       about  installation  such  as  Installation
       Class,  Id code, Area and their  physical
       and  tactical characteristics.

     member-attribute :
          Iclass
               Description : Installation class
               Value class : INS-CLASS
               Mandatory
               Not changeable
          Iid
               Description : Installation
                             Identification
               Value class : INS-ID
               Mandatory
               Not changeable
          Iarea
               Description : Estimated Area
               Value class : AREA
          Ipers
               description : Estimated persons
               Value class : NO-OF-PERSONS
          Iff
               Value class : FRIEND-OR-FOE
          Iclass/Itype
               description  :  Concatenation  of
                            weapon class and type
               Value class : PREC
               Match : PCLASS/PTYPE of PREC on PID
               'Cond2': Multivalued
                        Not changeable

     Class attribute :
            Total-foe
               description :  total  foe  numbers
               Value class : TOTAL-NUMBER

Identifier : Iid
```

Figure 4.2   SDM of IREC in the Intellience Database

C. SDM FOR INTELLIGENCE DATABASE

Figure 4.2, Figure 4.3, Figure 4.4, Figure 4.5 show a SDM logical schema for the Intelligence Datsabase. The data given in Appendix A to be used is composed of four records; First, Installation records which describe the normal Master file of Installation includes several fields such as Iclass, Iid, Iarea, Ipers, Iff, Iclass/Itype. Second, Ammunition records which describe all information about Ammunition include several fields such as Acat, Albs, Akill, Awar. Third, Photo records which describe all information of Photo taken includes several fields such as Pday, Pid, Pclass/Ptype, Pnum, Pwc. Finally Weapon records which have all information of Weapons include fields such as Wclass, Wtype, Wff, Wammo, Wrange, Wfeul, Wlbs. INSTALLATION is first defined. The class is named, and then an informal description of the class is provided. The description, which is optional, defines the purpose and content of the class. Special remarks are written here. Next, the member attributes are defined. These are attributes of the entities in this class. According to the Photo days in Photo record, Installation records are updated, so Iclass/Itype has Match function; Match : PCLASS/PTYPE of PREC on PID. And Ammunition and Weapon records are automatically updated.

(5) There are several primitive ways of finding interclass connections and derived attributes, corresponding to the most common types of information redundancy appearing in database applications. These facilities integrate multiple ways of viewing the same basic information.

1. Basic Format of an SDM Entity Class

The basic format of an SDM entity class description is given in Figure 4.1. **[Ref. 3]**

```
----------------------------------------------------------
|                                                        |
|   ENTITY-CLASS-NAME                                    |
|                                                        |
|       [description ------------]                       |
|       [interclass connection --------]                 |
|                                                        |
|          member attribute :                            |
|                                                        |
|             Attribute-name                             |
|                                                        |
|                value class : ---------                 |
|                [mandatory]                             |
|                [multivalued] [no overlap in values]    |
|                [exhaust value class][not changeable]   |
|                [inverse : Attribute-name]              |
|                [match : Attribute-name                 |
|                  ENTITY - CLASS on Attribute-name2]    |
|                [derivation : -----------]              |
|                                                        |
|      [ class attribute :                               |
|                                                        |
|          Attribute-name                                |
|              [description : -------]                   |
|              value class : --------                    |
|              [derivation : --------]]                  |
|                                                        |
| identifier : attribute-name + [Attribute-name2 + [ ]]] |
|                                                        |
----------------------------------------------------------
```

Figure 4.1 Format of SDM Entity Class Description

V. RELATIONAL DATABASE DESIGN

A. INTRODUCTION

The relational model was first proposed by Dr. E. F. Codd in a seminal paper in 1970 **[Ref. 13]**. This innovation stressed the independence of the relational representation from physical computer implementation such as ordering on physical devices, indexing, and using physical access paths. The model thus formalized the separation of the user view of data from its eventual implementation; it was the first model to do so. In addition, Codd proposed criteria for logically structuring relational databases and an implementation-independent language to operate on these databases. The relational model represents data in the simple form of tables. The relational model is attractive in database design because it provides formal criteria for logical structure, namely, normal form relations.

1. Terminology

A relation is simply a two-dimensional table that has several properties. First, the entries in the table are single-valued; neither repeating groups nor arrays are allowed. Relations are flat files. Columns of a relation are refered to as attributes. Each row of the relation is known as a tuple. If the relation has n columns, then each row is refered to as an n-tuple. Also, a relation that has n columns or n attributes is said to be of degree n.

2. Keys of Relation

This key is the attribute or set of attributes that uniquely identifies tuples in a relation. A relation key is formally defined as a set of one or more relation attributes concatenated so that the following three properties hold for all time and for any instance of the relation:

1. Uniqueness : The set of attributes takes on a unique value in the relation for each tuple.
2. Nonredundency : If an attribute is removed from the set of attributes, the remaining attributes do not posses the uniqueness property.
3. Validity : No attribute value in the key may be null.

When two or more attributes or attribute collections can be keys, they are called candidate keys. When one of the candidates is selected to be the key, it is called the primary key. When an attribute in one relation is a key of another relation, the attribute is called a foreign key. Foreign keys are important when defining constraints across relations.

3. Relational Algebra

The relational algebra consists of a set of relational algebra operators. Each operator has one or more relations as its input and produces a relation as its output. The three basic relational algebra operations are SELECTION, PROJECTION, and JOINING.

The SELECTION operator selects all tuples from some relation such that some attributes in each tuple satisfies some condition. A new relation, which contains the selected tuples, is then created.

The PROJECTION operator constructs a new relation from some existing relation by selecting only attributes of the existing relation and eliminating duplicate tuples in the newly formed relation.

The JOINING is a method of combining two or more relations into a single relation. At the outset it requires the choice of attributes to match tuples in the relations. Tuples in different relations, but with the same value of matching attributes, are combined into a single tuple in the output relation. The examples of using three basic operators will be shown in Chapter IV.

B. RELATIONAL NORMAL FORMS

Not all relational database designs are equal; some are better than others. Obviously, a design that meets the users´needs is better than one that does not, but there are other criteria as well. With some relations, changing data can have unexpected consequences. These consequences, called modification anomalies, are undesirable. These anomalies can be eliminated by changing the database design. Usually relations without modification anomalies are prefered. Some relations are independent, others are interdependent. Generally, but not always, the less interdependency, the better.

1. Modification Anomalies

Consider Ammunition relation in Figure 5.1. It has the attributes ACAT, ALBS, AKILL, and AWAR. The meaning of a tuple is that given an Ammo category, Weight of One round and Killing Radius and Warhead Category are determined.

For the data in Figure 5.1, if we delete the tuple for ACAT A, we will lose not only the fact that Ammo Category A's Weight is 410 lbs, but also the fact that Killing radius is 100 feet. This is called a DELETION ANOMALY; we may be losing more information than desired. We lose facts about three attributes with one deletion. This characteristic may be considered undesirable because it is usually unintended.

AMMUNITION (ACAT, ALBS, AKILL, AWAR)
Key : ACAT

ACAT	ALBS	AKILL	AWAR
A	410	100	1
B	175	5	3
C	510	150	1
D	950	500	4

Figure 5.1 The Ammunition Relation

Also, suppose we want to enter the fact that ACAT E has a killing radius of 525 feet. We can not enter this data into the Ammunition relation until a ACAT has ALBS and AWAR. This restriction seems unnecessary. This situation is called an Insertion Anomaly. We gain facts about three attributes with one insertion; or, stated negatively, we cannot insert a fact about one attribute until we have an additional fact about another attribute. These anomalies can be eliminated by the creating two new relations via projection. An example of this will be shown in Figure 5.9.

2. Classes of Modification Anomalies

There are many different types of modification anomalies. In the 1970s relational theorists chipped away at these types. Someone would find an anomaly, classify it, and think of a way to prevent it. This process generated improved criteria for designing relations. These criteria are called Normal Forms.

Codd, in his paper [Ref. 13] defined first, second, and third normal forms. Later, Boyce-Codd normal form was postulated, and then fourth and fifth normal forms were defined. As seen in Figure 5.2, each of these normal forms contains the other. A relation in fifth normal form is automatically in 1, 2, 3, BC, and 4 normal forms.

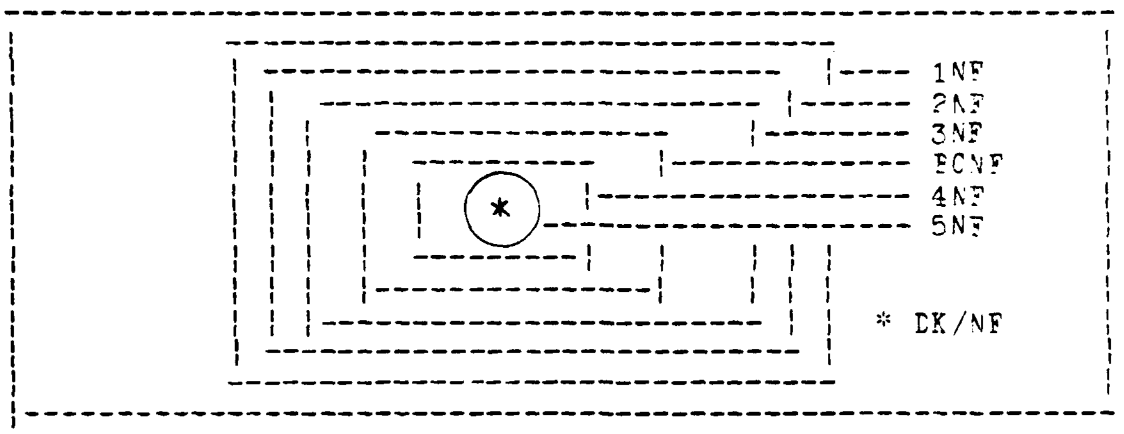


Figure 5.2 Relationship of Normal Forms

These normal forms were helpful, but they had a serious limitations. No theorist was able to guarantee that any of these forms would eliminate all anomalies; each form would eliminate just certain anomalies. This situation changed, however, in 1981 when R.Fagin defined a new normal form called DOMAIN/KEY normal form(DK/NF). Fagin showed that a relation in domain/key normal form is free of all modification anomalies, regardless of their types.

Until DK/NF was identified, it was necessary for relational database designers to continue looking for more and more anomalies, and more and more normal forms. Fagin's proof, however, greatly simplified the situation. If we can put a relation in DK/NF, then we are guaranteed it will have no anomalies.

3. Kinds of Normal Forms

All relations are in first normal form. A relation is in first normal form if and only if all underlying domains contain atomaic values only. Relations in first normal form have modification anomalies. It is possible to eliminate some of these anomalies by putting the relation in second normal form. We can eliminate even more when the relation is put in third normal form, and even more with Boyce-Codd normal form.

A functional dependency (FD) **[Ref. 6]** is a term derived from mathematical theory; it concerns the dependency of values of one attribute or set of attributes on those of another attribute or set of attributes. Formally, a set of attributes X is functionally dependent on a set of attributes Y if a given set of values for each attribute in Y determines a unique value for the set of attributes in X. The notation Y --> X is often used to denote that X is functionally dependent on Y. The att ibutes in Y are known as the determinant of the functional dependency Y --> X.

A relation is in second normal form if and only if it is in 1NF and every nonkey attribute is fully dependent on the primary key.

A relation is third normal form if it has the following properties: (1) The relation is in second normal form. (2) Every nonkey attribute is nontransitively dependent on the primary key.

A relation is in BCNF if every determinant is a candidate key. Since relations in BCNF have no anomalies regarding functional dependenies, this seemed to put the issue of modification anomalies to rest. However, it was soon discovered that anomalies can arise from situations other than functional dependencies.

Formally, multivalued dependency is defined as follows; In relation R(X,Y,Z), X ==> Y if each X value is associated with a set of Y values in a way that does not depend on the Z values.

A relation is in fourth normal form [Ref. 6] if it is in BCNF and has no multivalued dependencies. This definition means that if a relation has multivalued dependencies and is in fourth normal form, then the multivalued dependencies have a single value. In other words, all independent attributes have a single value.

A relation is in fifth normal form if and only if every join dependency in a relation is implied by the candidate keys of the relation.

A relation is in DK/NF if every constraint on the relation is a logical consequence of the definition of keys and domains. A constraint is any rule on static values of attributes that is precise enough that we can evaluate whether or not it is true. Thus intra- and inter-relation constraints, functional dependencies, multivalued dependencies, and join dependencies are all examples of constraints. DK/NF means that if we can find a way to define keys and domains such that all constraints will be satisfied when the key and domain definitions are satisfied, then modification anomalies are impossible. Unfortunately, there is no known way to convert a relation to DK/NF automatically, nor is it even known which relations can be converted to DK/NF. In spite of this, DK/NF can be exceedingly useful for practical database design.

C. RELATIONAL DATABASE DESIGN CRITERIA

Berri and co-workers [Ref. 9] have identified three relational criteria:

(1) Representation : The final structure must correctly represent the original specification.

(2) Separation : The original specifications are divided into relations that satisfy certain conditions.

(3) Redundancy : The final structure must not contain any redundant information.

First of all, the database must be separated into a number of normal form relations. The other two criteria are relatively general. In speific terms each can be applied to attributes, functional dependencies or data. To determine the criteria more specifically, notation for a relation and the input and output of a design process is needed.

A relation is defined as made up of two components, the attribute and the functional dependencies(FD) between the attributes. The definition takes the form

R = ({A,B,C}, {A ==>B, A ==> C})

Here R comprises three attributes, A, B, and C. The FDs between these attributes are A ==> B and A ==> C. The notation used to describe the input and output of the design process is Sin and Sout. Both Sin and Sout is are sets of relations. Here Sin is the input to the design process and Sout is the output. Most theoretical work is based on the universal relation assumption and assume that Sin is one relation, the universal relation, which is defined by a set of attributes and FDs, using the preceding notation, and that Sout is a set of normal relations, each of which is made up of a set of attributes and a set of FDs.

1. Satisfying Representation Criteria

One goal of any design process is to produce an output design, Sout, to accurately represent Sin. Further, all the relations in Sout must satisfy the conditions for normal form. C.Berri and co-workers(1978) [Ref. 9] have defined three representation criteria for the representation of Sin by Sout:

* REP1 : The relation Sout contains the same attributes as Sin.
* REP2 : The relation Sout contains the same attributes and the same FDs as Sin.
* REP3 : The relations in Sout contain the same attributes and the same data as Sin.

REP1 is trivial. It requires all the attributes in Sin to also apppear in the relations in Sout. But it does not consider any dependencies between the attributes.

In regard to REP2, recall that Sin is defined as a set of attributes and FDs and that each relation in Sout will also contain a set of attributes and a set of FDs. Representation REP2 requires that each FD in Sin be either

* contained as an FD in one of the relations in Sout or
* derived from the FDs in the relations in Sout, using the FD inference rules.

For example, in Figure 5.3, Sin = ({A,B,C}, {A ==> B, C ==> B}), Sout = (R2,R3) where R2 = ({A,B}, {A ==> B}) and R3 = ({B,C}, {C ==> B}). Thus R2 and R3 constitute the decomposition by projection of Sin.

A	B	C
a1	b1	c1
a3	b1	c2
a2	b2	c3
a4	b2	c4

DECOMPOSE

A	B
a1	b1
a3	b1
a2	b2
a4	b2

B	C
b1	c1
b1	c2
b2	c3
b2	c4

JOIN

A	B	C
a1	b1	c1
a1	b1	c2
a3	b1	c1
a3	b1	c2
a2	b2	c3
a2	b2	c4
a4	b2	c3
a4	b2	c4

Figure 5.3 Decomposition

2. Lossless Decompositions

Formally, a lossless decomposition can be described as follows. The decomposition of a relation R(X,Y,Z) into R1 and R2 is defined by two projections:

* R1 = projection of R over X,Y
* R2 = projection of R over X,Z

where X is the set of common attributes in R1 and R2. The decomposition is lossless if R = join of R1, R2 over X. The composition is lossy if R $\subset$ join of R1,R2 over X.

3. Redundancy Critera

Redundancy criteria can be defined in various ways. One way of defining redundancy criteria is as follows:

* RED1 : A relation in Sout is redundant if its attributes are contained in the other relations in Sout.
* RED2 : A relation in Sout is redundant if its FDs are the same or can be derived from the FDs in the other relations in Sout.
* RED3 : A relation in Sout is redundant if its content can be derived from the contents of other relations in Sout.

Obviously, RED1 is not a very useful criterion, because during separation it is often necessary to create separate relations that represent FDs between attributes, which may appear in other relations. On the other hand, RED2 and RED3 can be quite useful criteria. Any design algorithms should in particular avoid RED3, because it would keep the same data in more than one relation. Such relations could all be in normal form and no anomalies would occur in relations. However, interrelational a omalies would arise if some fact were updated in one relation but the other. Designs that include RED2 would cause the same problem.

4. Elimination of Modification Anomalies

If relations can be put into DK/NF, then no modification anomalies can occur. Thus DK/NF becomes a design objective, and relations that are in DK/NF are usually preferred.

Not all relations, however, can be put into DK/NF. This occurs when there are constraints that cannot be expressed as logical consequences of keys and domains. As example described by Fagin **[Ref. 14]** is a relation having the following constraints: The relation must never have fewer than three tuples. There is no way to express this constraint in terms of domains and keys. Thus it has a modification anomaly. In fact, this strange relation has a deletion anomaly but no insertion anomaly.

When relations cannot be tranformed into DK/NF, the constraint that cannot be expressed in terms of domains and keys must be inserted into application programs. This is undesirable because the constraint is hidden.

5. Ease of Use

A fifth criterion for a relational design is ease of use. As far as possible, we strive to structure the relations so that they are familiar and seem natural to users. Sometimes this goal conflicts with the elimination of anomalies or with independence.

D. RELATIONAL DATABASE MANAGEMENT SYSTEM

This section [Ref. 3] describes the relational model as the implementation model that is supported by a DBMS. Any relations produced during data analysis can be implemented directly on this DBMS. Because of its tabular interface, the relational model makes an attractive implememtaion model. It is receptive to two types of environments:

* the traditional data processing environment, in which databases are set up by professional computer programmers on behalf of database users.
* environments in which nonprogrammer users set up their own databases.

The relational model provides the same advantages in both types of environments. Its natural interface simplifies the design and use of the database. This is particularly so if a language with powerful selective capabilities can be provided by the DBMS. Such languages can reduce program development time and hence are attractive in commercial data-processing environments. They are also attractive to

VI. IMPLEMENTATION USING ORACLE

The Intelligence database has been implemented using the ORACLE relational DBMS. Initially a data file is created using CREATE command. After the creation of the IREC file, it appears as shown below.

```
UFI> CREATE TABLE IREC
  2      (ICLASS  CHAR(2),
  3       IID     NUMBER(5),
  4       IAREA   NUMBER(2),
  5       IPERS   NUMBER(4),
  6       IFF     CHAR(3));

Table created.
```

After the table is created, IREC data is added to the data file using the INSERT command.

```
UFI> INSERT INTO IREC VALUES ('AF',101,8,1500,'FOE');

1 record created.

UFI> INSERT INTO IREC VALUES ('AF',110,10,1800,'FOE');

1 record created.
```

After IREC file is created, list all the data in the IREC using SELECT, FROM command.

```
UFI> SELECT *
  2  FROM IREC;

IC        IID   IAREA    IPERS IFF
-- -------- ------- -------- ---
AF      101       8     1500 FOE
AF      110      10     1800 FOE
PO      208      25     4600 FRD
AR      318       3     2800 FOE
AR      303       1      900 FOE
PO      215      32     3900 FOE
AF      109       7     1400 FRD
PO      223      35     5200 FOE
AR      316       5     3800 FRD
PO      231      30     7500 FRD

10 records selected.
```

areas as concurrency, locking, security, integrity, view definition, etc, has taken the relational approach as a starting point, precisely because it provides a clean conceptual base. As for the question of an undering theory, the realtional approach is not only soundly based on certain aspects of mathematical set theory, but it also possesses a considerable body of theory in its own right aimed specifically at its application to database problems.

In a relational schema the entire information content of the database is represented by means of a single data construct, namely, the n-ary relation. In a network schema, by contrast, there exits at least one fanset bearing information essentially; for it there did not, the schema would degenarate into a relational schema with certain explicit access paths. In other words, there are at least two essential data constructs in the network approach, the baseset and fanset. In DBTG, in particular, there are five data constructs, any or all of which may be used to describe essential information:

* record type (corresponds to baseset);
* DBTG set (corresponds to fanset);
* singular set;
* ordering;
* repeating group.

D. A COMPARISON WITH THE NETWORK APPROACHES

Successful DBTG systems lack the flexibility of relational systems, but they make up for it in being able to process larger amounts of data more quickly. Systems like this excel at standardized, repetitive applications such as online teller processing, or large-scale order entry, and the like. They may not be elegent, but they can do large amounts of work, and do it well.

Thus, we have the following situation: relational systems are easy to use, applications can be quickly developed, but processing of very large amounts of data can be unacceptably slow. On the other hand, DBTG is more difficult to use, but large amounts of work can be quickly and efficiently accomplished. The DBTG representation of the Intelligence Database is given in Appendix B.

These observations were true in 1983, but development efforts are underway in both camps to eliminate the shortcomings. Vendors of relational systems are striving to improve performance, whereas vendors of nonrelational systems are attempting to make their systems easier to use. One way they are doing this is to give the nonrelational systems a relational appearance to the user.

In the relational approach, all information in the database is represented using one construct, and moreover this one construct is both simple and familiar. It is significant that most of the research since 1970 into such

4. Attribute Domains

Each attribute has its own domain. The value of each attribute must be within its domain. The domain of each attribute is shown in Figure 5.9.

Attribute	Domain
WCT	WEAPON-CLASS + WEAPON-TYPE
WFF	FRIEND-OR-FOE
WRANGE	RANGE
WFUEL	FUEL-CAP
WLBS	MAX-LOAD
ICLASS	INS-CLASS
IID	INS-ID
IAREA	AREA
IPERS	NO-OF-PERSONS
IFF	FRIEND-OR-FOE
PDAY	DATE
PID	INS-ID
PCT	WEAPON-CLASS + WEAPON-TYPE
PNUM	NUM-OF-WEP
PWC	WEATHER
ACAT	AMMO-CATEGORY
ALBS	MAX-LOAD
AKILL	RANGE
AWAR	WARHEAD-CAT

Figure 5.9 Attribute Domains

3. Relations of Intelligence Schema

After these four records are examined, Inverse and Match functions must be deleted in order to achieve DK/NF. We have repeating groups, because IREC and WREC have multiple values. Repeating groups, however, are prohibited in relational databases, so two inte relation constraints, AW and IDTEMP, were added. The AW record is composed of WCLASS, WTYPE, and ACAT; and IDTEMP is composed of IID, ICLASS, and ITYPE.

Because of interrelation constraints, Weapon class and Weapon Type are omitted from IREC, and Wammo is omitted from WREC. All attributes are dependent on the primary key, so there are no modification anomalies. The relations in the INTELLIGENCE Schema is given in Figure 5.9.

```
        WREC (WCLASS, WTYPE, WFF, WRANGE, WFEUL, WLBS)
             key : WCLASS + WTYPE
        IREC (ICLASS, IID, IAREA, IPERS, IFF)
             KEY : IID
         PREC (PDAY, PID, PCLASS, PTYPE, PNUM, PWC)
             Key : PDAY + PID + PCL SS + PTYPE
         AREC (ACAT, ALBS, AKILL, AWAR)
             KEY : ACAT
         AW (WCT, ACAT)   ---
                            |  Interrelation Constraints
         IDTEMP (IID, WCT) --
```

Figure 5.9 The Relations in the Intelligence Schema

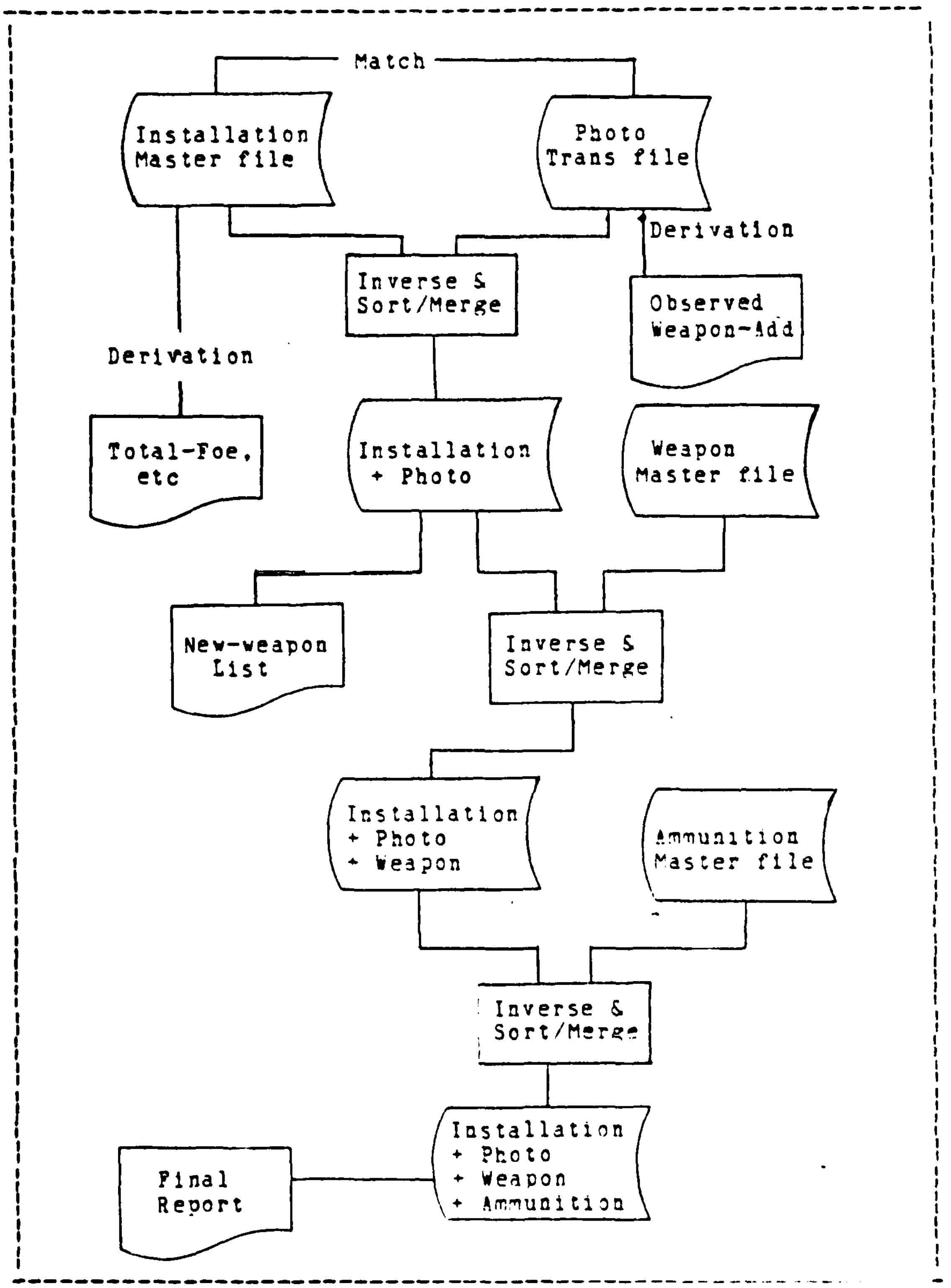


Figure 5.8 System Flowchart and Relationships

2. System Flowchart and Relationships

The system flowchart and relationships between the various master files are shown in Figure 5.8. It shows how all four master files can be updated automatically by utilizing the Photo master files. The use of simple query language will produce a large volume of new data easily and quickly. Three SDM facilities will be used to explain how it works.

Initially, the four maste files are created. Installation records are sorted according to the IID, and Photo records are sorted accordin to the PID. The derivation facility will yield the Total-foe-nnmber from the Installation file and the Observed-weapon-add from the Photo file. The inverse facility on the two master files yields the new master file called Installation and Photo file which includes PNUM and PWC. The derivation facility will produce the new-weapon list from the Installation and Photo file by comparing PDAY with previous PDAY. The inverse facility on this new master file and the Weapo master file will yield the new master file called Installation and Photo and Weapon by comparing WCLASS and WTYPE with ICLASS and ITYPE giving us new information such as WAMMO, WRANGE, WFEUL. The final use of IREC, PREC, WREC, AREC files, necessitated by repeating WAMMO groups, yields the new master file called Installation, Photo, Weapon, and Ammunition giving us the new information such as ALBS, AWAR.

implemented, because security authorizations will relate to relations.

D. CONVERSION OF SDM INTO RELATION DATABASE DESIGN

1. Relationship Between Records

The relationships for the Intelligence Database are given in Figure 5.7. Inversion, Matching and Derivation will be used to provide inter-relationships between the attributes shown. It is possible to find duplicated field names using these methods.

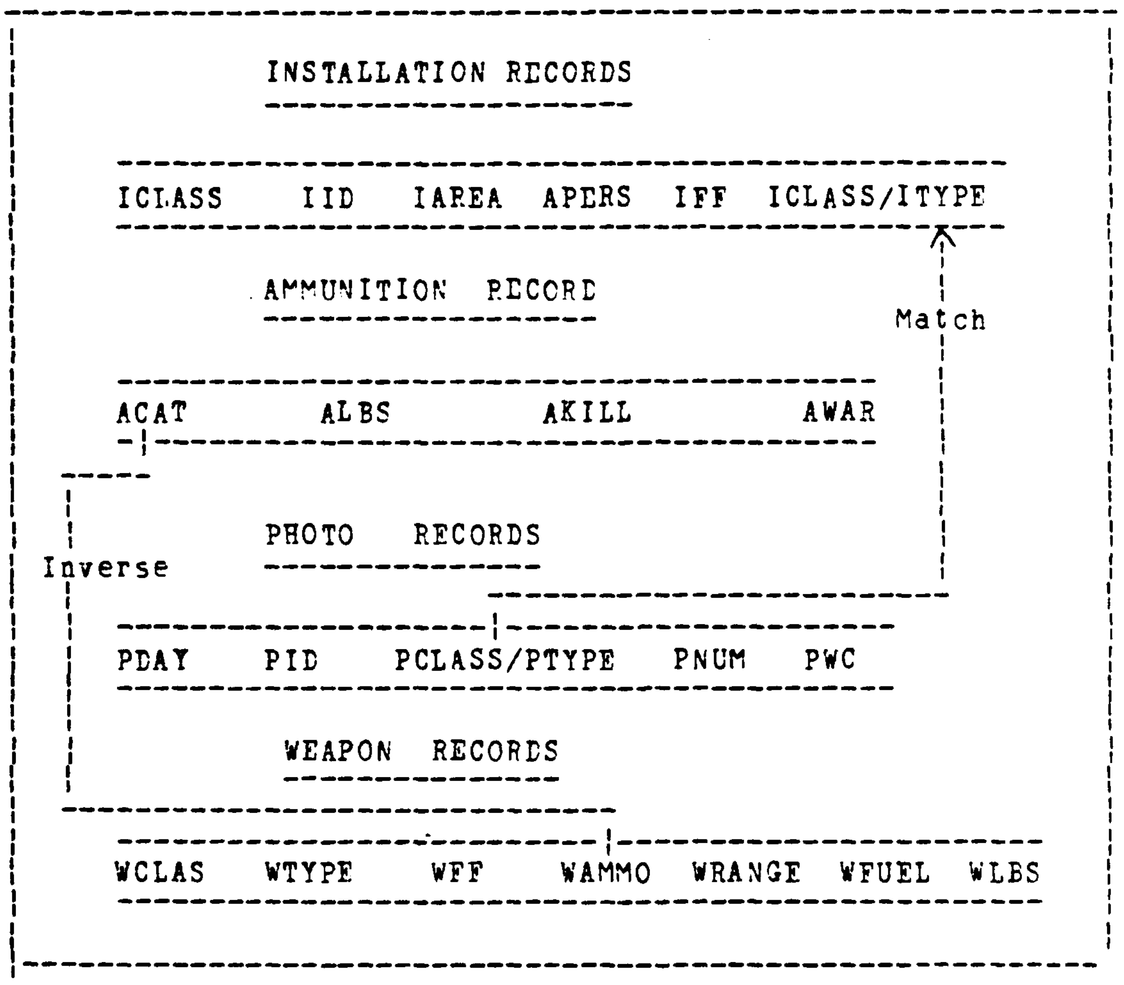

INSTALLATION RECORDS
ICLASS IID IAREA APERS IFF ICLASS/ITYPE
AMMUNITION RECORD
Match
ACAT ALBS AKILL AWAR
PHOTO RECORDS
Inverse
PDAY PID PCLASS/PTYPE PNUM PWC
WEAPON RECORDS
WCLAS WTYPE WFF WAMMO WRANGE WFUEL WLBS

Figure 5.7 The Relationships between Records

the JOIN operation is likely to take substantial machine time. It may be feasible with small relations, but some commercial files are hundreds of million of bytes long. In understanding the performance issue, it is very important to remember that the relations and the operations on them such as the JOIN will never take place physically. Instead, equivalent results will be produced by means of pointer structures or indices.

A relational database design is sometimes depicted as not being 'driven' by a user view of the data. A new unanticipated user view can be handled with ease if the data it needs are stored. Although this is true in connection with the logical structure of the data, the new view may not be handled with good machine performance because the physical structure of the data was designed to best serve the most common applications. The physical structure is user-driven even if the logical structure is not.

The advantage of relational database is first of all, ease of use. That means the easiest way to represent most data is with two dimensional tables. Another advantage is flexibility. Users can use PROJECTION and JOIN in the form they want. Another advantage is precision. This means that the precise results of relational mathematics can be applied to the manipulation of relations. Computer security is another important application area where the relational model should be considered. Security controls can be easily

5.5, the output of the precompiler is then input to a standard languge compiler for compilation in normal fashion.

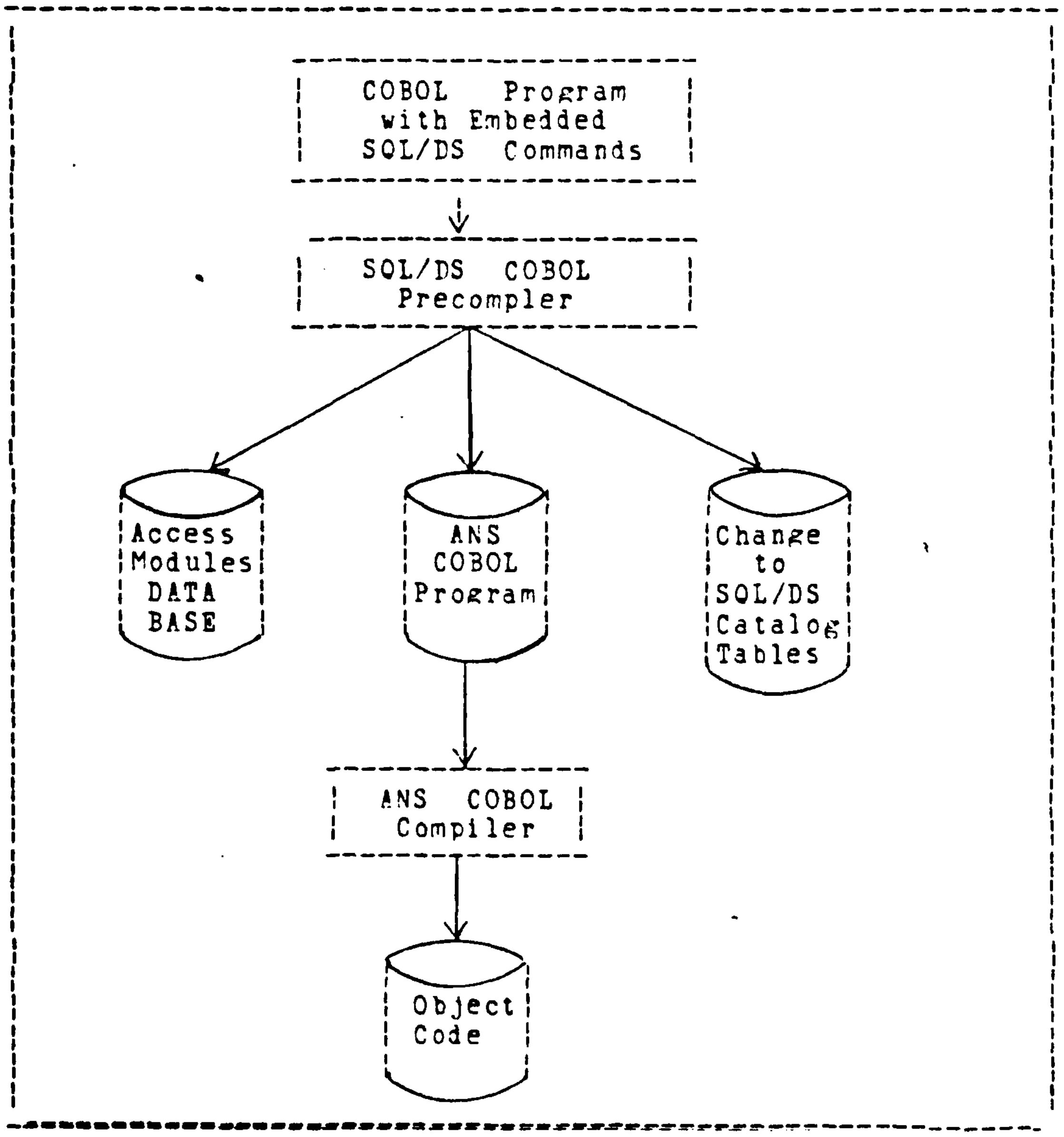


Figure 5.5 Role of SQL/DS Precompiler

4. Advantage and Disadvantage of Relational Database

A disadvantage sometimes cited for a relational database is machine performance. With present-day hardware update activities. No application programming is required when using ISQL. For this type of access, users must be connected to a communications control program such as CICS or equivalent.

A second mode of access is via application programs. In this mode, SQL/DS commands are embedded in standard programming text like COBOL, PL/1, or assembler language. These embedded commands are nearly identical to the commands that are issued to ISQL. This means that application programmers need learn only one data language; the single data language can be used from application programs or interactively with ISQL. Users claim the near identity between ISQL statement and embedded SQL/DS statements helps them to develop application programs. Programmers can develop database commands interactively, verify them for correctness using ISQL, and then include those commands in application programs.

Figure 5.5 shows the processing of embedded SQL/DS statements. Programs containing SQL/DS commands are input to a precompiler that examines the statements for correctness and builds small SQL/DS access modules that will perform the desired database service. These modules are stored in the database. At the same time, program instructions are inserted into application programs to call the stored access modules when needed. The precompiler generates these instructions in standard COBOL or PL/1. As shown in figure user to process data without concern for physical data structures.

There are many other relational DBMS. Figure 5.4 lists some of the major systems as of late 1982. There is also a microcomputer relational product: dBASE II, which operates on CP/M-based micro. dBASE II is an example of a relational (or tabular) DBMS that restricts join operations. The join columns must be indexed.

SQL-Based System

- SQL/DS, IBM
- ORACLE , Relational Software, Inc.
- System R, IBM

QUEL-Based Systems

- INGRES, Relational Technology, Inc
- IDM 500, Britton-Lee,Inc

Other Relational Systems

- MRDS/LINUS, Honeywell
- dBASE II, Ashton-Tate
- NOMAD, National Compute Sharing Services

Figure 5.4 Relational DBMS Products and Vender

3. Two Modes of Access of SQL/DS

SQL/DS can be used either interactively from a terminal or via application programs. The interactive processor, ISQL, processes SQL commands to perform query and

2. Commercial Relational DBMS

There are currently many commercial DBMS products that claim to be relational. Some are more relational in name than in actuality. Criteria can be used to assess whether or not a product is truly a relational product. Specially, the DBMS should model data as tables, and it should support SELECT, PROJECT, and unrestricted JOIN operations.

Relational DBMS can be divided into three groups. One group is based on the data language SQL, one on the data language QUEL, and a group that contains systems falling into neither of these categories.

Three major SQL-based DBMS products are SQL/DS, System R, and ORACLE. System R is a research system developed by IBM for the study of relational technology. ORACLE is vended by Relational Software Incorporated. Originally, ORACLE was developed for operation on Digital Equipment Corporation PDP minicomputers. Since its origin, ORACLE has been converted to operate on IBM mainframes as well. ORACLE's user interface is based on SEQUEL II, an earlier version of SQL. According to RSI, ORACLE will soon be compatible with the current version of SQL. QUEL is a data language like SQL. (Just like COBOL and PL/I are alternative programming languages, SQL and QUEL are alternative data languages.) QUEL is based on tuple relational calculus. QUEL is nonprocedual and allows the nonprogrammer users, allowing them to use the database without resorting to computer-oriented procedual languages.

1. Relational Characteristics

What characteristics must a DBMS have to be considered a relational product? In his Turing lecture, E.F Codd [Ref. 15] defined a relational DBMS as one in which data is defined in tables and processed by using SELECT, PROJECT and unrestricted JOIN operations, or their equivalent. Codd called a system having these characteristics MINIMALLY RELATIONAL.

SELECT, PRODUCT, and JOIN will be used in Chapter VI. The SELECT obtains rows of the table according to criteria on row contents. PROJECT obtains columns of a table by column name. Finally, JOIN brings two relations together based on the relationship between two columns having the same domain.

Some DBMS products specify that only columns can be used as JOIN criteria. For example, a DBMS may require the columns used as JOIN criteria to be indexed. This implies the undesirable situation of restricting user activity because of physical data re resentation. To the nonspecialist user, this restriction appears arbitrary. In fact, there is no logical reason for this restriction; it exists only to improve performance. To eliminate this situation, Codd specifies that a minimally relational system must have unrestricted JOINS. This means that any column can be used as criteria for the JOIN.

In the same way, data for the other relations are created and are shown below. List all AREC file.

```
UFI> SELECT *
  2  FROM AREC;

A     ALBS   AKILL    AWAR
- ------- ------- -------
A     410     100       1
B     175       5       3
C     510     150       1
D     950     500       4
E    1100     525       4
F    1300     600       5
G       8       1       2
H     125       2       6
I      12       1       2
J     180     100       7
K     240     125       8
L    1450     400       9
M    1300     500       9
N     150       2       9
P     150       1      10

15 records selected.
```

List all WREC file.

```
UFI> SELECT *
  2  FROM WREC;

WCL    WTYPE WFF  WRANGE   WFUEL    WLBS
--- ------- --- ------- ------- -------
AC         1 FDE   10000     800   10000
AC         2 FDE    8000     700   15000
AC         3 FDE    5000     500   11000
AC         4 FRD    9000     600   11000
AC         5 FRD   11000     800   15000
AC         6 FRD    5000     700   12000
SH         1 FDE   30000    5000  100000
SH         2 FDE   25000    7000  125000
SH         3 FDE   15000    6000  110000
SH         4 FRD   35000    7000  115000
SH         5 FRD   20000    8000  130000
SH         6 FRD   12000    6000  110000
ARU        1 FDE    3500     500    5000
ARU        2 FDE    1000     200    2500
ARU        3 FDE    3000     300    4000
ARU        4 FRD    3000     600    6000
ARU        5 FRD    1000     250    3000
ARU        6 FRD    2500     300    4500

18 records selected.
```

List all PREC file.

```
UFI> SELECT *
  2  FROM PREC;

      PDAY        PID PCL    PTYPE       PNUM PWC
------- -------- --- -------- -------- -----
       301        318 ARU          1        100 FAIR
       301        318 ARU          3        200 FAIR
       301        316 ARU          5        150 FAIR
       302        110 AC           1          7 PCLDY
       302        208 SH           5          4 FAIR
       302        101 AC           2          8 PCLDY
       302        215 SH           3         25 PCLDY
       302        223 SH           3          8 PCLDY
       302        223 SH           1          4 PLCDY
       302        303 ARU          2        200 FAIR
       303        110 AC           1         10 CLDY
       303        223 SH           3          4 CLDY
       303        318 ARU          1        200 PCLDY
       303        231 SH           6         30 CLDY

14 records selected.
```

List all IDTEMP file.

```
UFI> SELECT *
  2  FROM IDTEMP;

     IID ICL   ITYPE
------- --- -------
     101 AC          1
     101 AC          3
     110 AC          1
     110 AC          1
     110 AC          2
     110 AC          3
     208 SH          4
     208 SH          6
     318 ARU         1
     318 ARU         3
     318 AC          3
     303 ARU         2
     215 SH          1
     215 SH          3
     215 AC          2
     108 AC          4
     108 AC          5
     223 SH          2
     223 SH          3
     223 AC          2
     316 ARU         4
     316 ARU         5
     316 AC          6
     231 SH          5
     231 SH          6
     231 AC          5

26 records selected.
```

List all AW file.

```
UFI> SELECT *
  2  FROM AW;

WCL   WTYPE A
--- ------- -
AC        1 A
AC        1 C
AC        2 B
AC        2 C
AC        3 B
AC        3 G
AC        4 K
AC        4 L
AC        5 K
AC        6 L
AC        6 P
SH        1 D
SH        1 E
SH        1 F
SH        2 E
SH        2 F
SH        3 D
SH        4 M
SH        5 N
SH        6 N
ARU       1 G
ARU       1 H
ARU       2 J
ARU       3 H
ARU       3 J
ARU       4 P
ARU       4 R
ARU       5 R
ARU       6 P
ARU       6 R

30 records selected.
```

Several sample queries and the results using ORACLE are given below.

1. List what kinds of Installation Classes are in IREC.

```
UFI> SELECT UNIQUE ICLASS
  2  FROM IREC;

IC
--
AF
PO
AR
```

2. List how many Installation ID Codes are in IREC.

```
UFI> SELECT COUNT(IID)
  2  FROM IREC;

COUNT(IID)
----------
        10
```

3. List Installation record file sorted by Installation ID Code in ascending order.

```
UFI> SELECT *
  2  FROM IREC
  3  ORDER BY IID;

IC      IID   IAREA   IPERS IFF
-- ------- ------- ------- ---
AF     101       8    1500 FOE
AF     108       7    1400 FRO
AF     110      10    1800 FOE
PO     208      25    4600 FRO
PO     215      32    3900 FOE
PO     223      35    5200 FOE
PO     231      30    7500 FRO
AR     303       1     900 FOE
AR     316       5    3800 FRO
AR     318       3    2800 FOE

10 records selected.
```

4. List how many Friends or Foes are in the Installation records where the IFF is equal to Foe.

```
UFI> SELECT COUNT(IFF)
  2  FROM IREC
  3  WHERE IFF = 'FOE';

COUNT(IFF)
----------
         6
```

5. For Installation ID Code 110, display the weapons (Class/Type) observed in the past at the installation, the Day of Photo and Number of Weapons observed which correspond

to those weapons observed in the past, ONLY for those weapons with a maximum ammo load in excess of 10,500 pounds.

```
UFI> R
  1  SELECT IDTEMP.ICLASS,IDTEMP.ITYPE,PREC.PDAY,PREC.PNUM
  2  FROM IDTEMP,PREC,WREC
  3  WHERE IDTEMP.IID = 110
  4      AND IDTEMP.ICLASS = PREC.PCLASS
  5      AND IDTEMP.ITYPE = PREC.PTYPE
  6      AND PREC.PCLASS = WREC.WCLASS
  7      AND PREC.PTYPE = WREC.WTYPE
  8*     AND WREC.WLBS > 10500

ICL   ITYPE    PDAY    PNUM
--- ------- ------- -------
AC        2     302       8
```

6. Display the Installation ID Code and Area for those installations photographed on Day 301 for which the weapons (Class/Type) observed on that day had a maximum range in excess of 7,000 meters, and the killing radius of all ammunition types available exceeds 125 feet.

```
UFI> R
  1  SELECT IREC.IID,IREC.IAREA
  2  FROM IREC,PREC,WREC,AW,AREC
  3  WHERE AREC.AKILL > 125
  4      AND WREC.WRANGE > 7000
  5      AND PREC.PDAY = 301
  6      AND IREC.IID = PREC.PID
  7*     AND AW.ACAT = AREC.ACAT

    IID   IAREA
------- -------
    318       3
    318       3
    315       5
```

7. Display Installation ID Code and the total number of weapons observed according to Installation ID Code,

weapon classes and weapon type.

```
UFI> SELECT PID,SUM(PNUM)
  2  FROM PREC
  3  GROUP BY PID,PCLASS,PTYPE;

      PID SUM(PNUM)
------- ---------
      101         8
      110        17
      208         4
      215        25
      223         4
      223        12
      231        30
      303       200
      316       150
      318       300
      318       200
```

8. Display Installation Class and Weapon class and Weapon Type and the total number of Weapons Observed, where Installation ID Code in INSTALLATION record is equal to that of PHOTO record together with Installation Class and Weapon class and Weapon type.

```
UFI> R
  1  SELECT ICLASS,PCLASS,PTYPE,SUM(PNUM)
  2  FROM IREC,PREC
  3  WHERE PID = IID
  4* GROUP BY ICLASS,PCLASS,PTYPE

IC PCL   PTYPE SUM(PNUM)
-- --- ------- ---------
AF AC        1        17
AF AC        2         8
AR ARU       1       300
AR ARU       2       200
AR ARU       3       200
AR ARU       5       150
PO SH        1         4
PO SH        3        37
PO SH        5         4
PO SH        5        30

10 records selected.
```

9. Display Pday of Photo for any day that Wrange is greater than 8000, Wlbs is greater than 10000 and Wfeul is 600, according to the information in the WREC record.

```
UFI>
UFI>
UFI> SELECT UNIQUE PDAY
  2  FROM WREC, PREC
  3  WHERE WRANGE > 8000
  4     AND WLBS > 10000
  5     AND WFUEL > 600
  6     AND WREC.WCLASS = PREC.PCLASS
  7     AND WREC.WTYPE = PREC.PTYPE;

   PDAY
-------
    302
    303
```

10. List all field names and its type for Photo record.

```
UFI> DESCRIBE PREC
  # size csize type                               name
  1   22     40  1 numeric                        PDAY
  2   22     40  1 numeric                        PID
  3    3      2  2 character                      PCLASS
  4   22     40  1 numeric                        PTYPE
  5   22     40  1 numeric                        PNUM
  6    5      2  2 character                      PNC
```

VII. CONCLUSIONS AND RECOMMENDATIONS

An Intelligence Database system is very complex and important, and needs very accurate information to increase war power.

Manual systems can not reduce national defense expenditures and make it difficult to obtain accurate information from the Intelligence system. Thus, database management systems must be used in Intelligence systems in order to increase end-user productivity, decrease staff, enable work to be done more efficiently, and permit end-user management more authority and responsibility.

Relational database models will be most useful in Intelligence systems, because this model gives structural independence for the database and a high level language for queries. Normal forms and query optimization techiniques can be applied to decrease inefficiency of the relational database model in the system design stage.

When we design a database, the SDM model is very important. SDM is a high-level semantics-based database description and structuring formalism for the database and enhances usability of the database system.

The output of SDM is a specification that can be used to implement the database using a commercial DBMS. The output of SDM has two alternatives. If we are going to use

DBMS based on the relational model, we will produce a relation design. If we are going to use a DBMS based on the CODASYL DBTG model, it will produce a DBTG design.

If we constructed an SDM model, it would be easy to reduce the effort required to convert elational models into DBTG models or vice versa.

Using the output of SDM in the Intelligence system, the records are rearranged in order to fit a relational model. (e.g., creation of the interrelational constraints). The ORACLE DBMS was used to demonstrate an operative relational DBMS. The ORACLE database management system is a good relational database model, providing a user friendly environment, easy to use and fast access to data.

It seems appropriate to conclude with Codd's statement of the objectives for the relational approach [Ref.12]. They are as follows:

1. To provide high degree of data independence.
2. To provide a community view of the data of spartan simplicity, so that a wide variety of users in an enterprise can interact with a common view (while not prohibiting superimposed user views for specialized purposes).
3. To simplify the potentially formidable job of the database administrator.
4. To introduce a theoretical foundation into database management.

5. To merge the fact retrieval and file management fields in preparation for the addition at a later time of inferential services in the commercial world.

6. To lift database application programming to a new level - a level in which sets (and more specially relations) are treated as operands instead of being processed element by element.

No one would claim that all these objectives have now been attained; much more work remains to be done. However, a strong foundation has been established, and there seems good reason to be optimistic about the eventual outcome.

APPENDIX A

ORIGINAL DATA

Four record types constitute the Intelligence Database attached. The following notes and definitions apply to the database.

1. The Installation, Ammunition and Weapon Records represent the status as of the end of day 300. The photo records represent information obtained on the indicated day (not neccessarily in addition to status information on day 300).

2. Defintions

Installation Class :	AF - airfields
	PO - ship ports
	AR - Army units

Weapon Class :	AC - aircraft
	SH - ship
	ARU - armour unit (eg., tank)

Weapon types are numbered 1, 2, 3, 4, 5, 6, for ea h class

Ammunition categories are letters A,B,C,D,E,F,G,

3. The occurence of the database as given is assumed to be indicative of the structure in the determination of unique keys, record relationships, functional dependencies, etc.

4. Variables have been given different names when they appear in different record types. (eg., IID and PID both refer to Installation ID Code).

5. There are cases where repeating group data is represented on the page of a particular record type. (eg., Weapon Class/Type with Installation Records).

INSTALLATION RECORDS (IREC)

ICLASS	IID	IAREA	IPERS	IFF	ICLASS/ITYPE
AF	101	8	1500	FOE	AC/1, AC/3
AP	110	10	1800	FOE	AC/1, AC/2,AC/3
PO	208	25	4600	FRD	SH/4,SH/6
AR	318	3	2800	FOE	ARU/1,ARU/3,AC/3
AR	303	1	900	FOE	ARU/2
PO	215	32	3900	FOE	SH/1,SH/3,AC/2
AF	108	7	1400	FRD	AC/4,AC/5
PO	223	35	5200	FOE	SH/2,SH/3,AC/2
AR	316	5	3800	FRD	ARU/4,ARU/5,AC/6
PO	231	30	7500	FRD	SH/5,SH/5,AC/5

ICLASS	: Installation Class
IID	: Installation Code
IAREA	: Area (Square Miles)
IPERS	: Estimated No. of Personnel
IFF	: Friend or Foe
ICLASS/ITYPE	: Weapon Class/Type Observed In Past

AMMUNITION RECORDS (AREC)

ACAT	ALBS	AKILL	AWAR
A	410	100	1
B	175	5	3
C	510	150	1
D	950	500	4
E	1100	525	4
F	1300	600	5
G	8	1	2
H	125	2	6
I	12	1	2
J	180	100	7
K	240	125	8
L	1450	400	9
M	1300	500	9
N	150	2	8
O	7	1	10

ACAT : Ammo Category

ALBS : Weight of One Round (Pounds)

AKILL : Killing Radius (Feet)

AWAR : Warhead Category

PHOTO RECORDS (PREC)

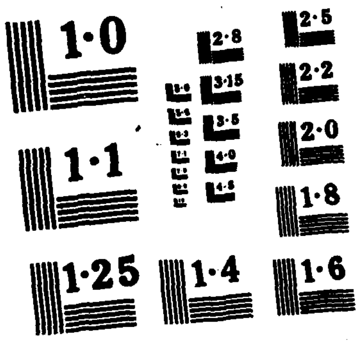
PDAY	PID	PCLASS	PTYPE	PNUM	PWC
301	110	AC	1	5	FAIR
301	110	AC	3	6	FAIR
301	208	SH	5	4	PCLDY
301	223	SH	3	6	PCLDY
301	223	SH	2	5	PCLDY
301	318	ARU	1	100	FAIR
301	316	ARU	5	200	FAIR
302	110	AC	1	7	PCLDY
302	208	SH	5	4	FAIR
302	101	AC	2	8	PCLDY
302	215	SH	3	25	PCLDY
302	223	SH	3	8	PCLDY
302	223	SH	1	4	PCLDY
302	303	ARU	2	200	FAIR
303	110	AC	1	10	CLDY
303	223	SH	3	4	CLDY
303	318	ARU	1	200	PCLDY
303	231	SH	6	30	CLDY

PDAY : Day of Photo
PID : Installation Code
PCLASS : Weapon Class
PTYPE : Weapon Type
PNUM : Number of Weapons Observed
PWC : Weather Condition

AD-A154 095   DESIGN AND IMPLEMENTATION OF AN INTELLIGENCE DATABASE   2/2
(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA J E JANG
DEC 84

UNCLASSIFIED   F/G 9/2   NL

END
FILMED
DTIC


1·0
2·8
2·5
3·15
2·2
3·5
2·0
1·1
4·0
4·5
1·8
1·25
1·4
1·6

WEAPON RECORDS (PREC)

WCLASS	WTYPE	WFF	WAMMO	WRANGE	WFEUL	WLBS
AC	1	FOE	A,C	10000	800	12000
AC	2	FOE	B,C	8000	700	15000
AC	3	FOE	B,G	5000	500	11000
AC	4	FRD	K,L	9000	600	11000
AC	5	FRD	K	11000	800	15000
AC	6	FRD	L,P	5000	700	12000
SH	1	FOE	D,E,F	30000	5000	100000
SH	2	FOE	E,F	25000	7000	125000
SH	3	FOE	D	15000	6000	110000
SH	4	FRD	M	35000	7000	115000
SH	5	FRD	M,N	20000	8000	130000
SH	6	FRD	N	12000	6000	110000
ARU	1	FOE	G,H	3500	500	5000
ARU	2	FOE	J	1000	200	2500
ARU	3	FOE	H,J	3000	300	4000
ARU	4	FRD	P,R	3000	600	6000
ARU	5	FRD	R	1000	250	3000
ARU	6	FRD	P,R	2500	300	4500

WCLASS : Weapon Class
WTYPE : weapon Type
WFF : Friend or Foe
WAMMO : Available AMMO Categories
WRANGE : Maximum Weapon Range
WFEUL : Feul Capacity (Gallons)
WLBS : Maximum Ammo Load (Pounds)

APPENDIX B

DBTG Schema for Intelligence Database

Figure B.1 presents a data structure diagram of the schema design for the Intelligence database. There are seven records and six sets. The names of the records and sets are shown in Figure B.1.

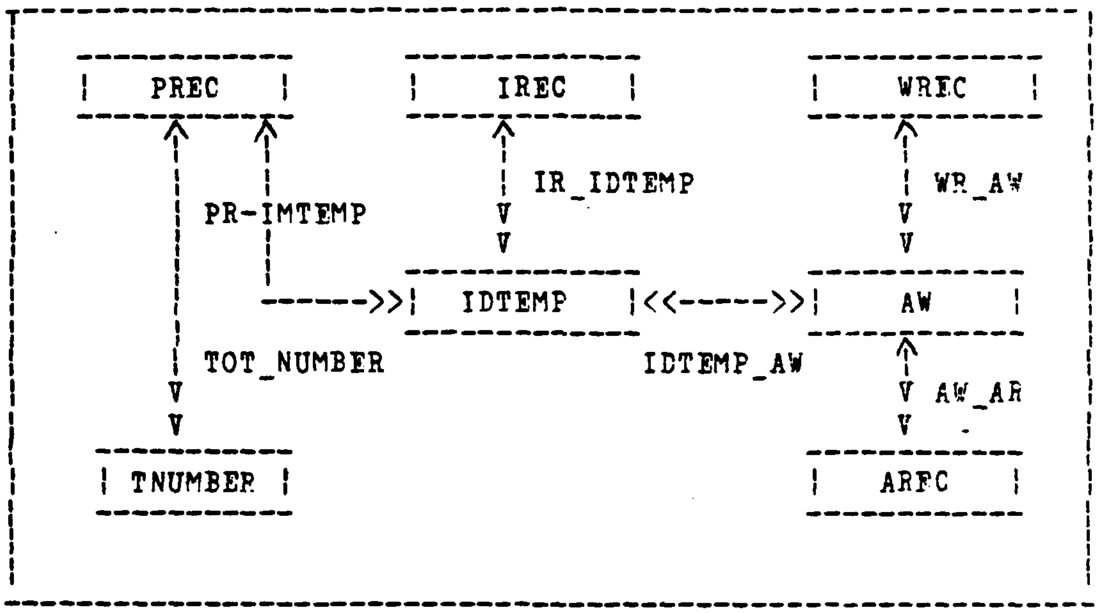


Figure B.1 DSD for Intelligence

Figure B.2 shows a schema description for Intelligence. This schema describes records, data-items and sets. According to the 1981 standard, no punctuation is required because keywords indicate the boundaries of phases and expressions.

```
SCHEMA name is Intelligence
     Record name is IREC
          duplicates are not allowed for IID
          ICLASS     type is   character   2
                     check is equal 'AF','AP','PC'
          IID        type is   fixed       3
          IAREA      type is   fixed       2
          IPERS      type is   fixed       4
          IFF        type is   character   3
                     check is equal 'FOE','FRD'
     Record name is PREC
          PDAY       type is   fixed       3
                     check is less than 366
          PID        type is   fixed       3
          PCLASS     type is   character   3
          PTYPE      type is   fixed       1
          PNUM       type is   fixed       3
          PWC        type is   character   5
     Record name is  AREC
          duplicates are not allowed for ACAT
          ACAT       type is   character   1
          ALBS       type is   fixed       5
          AKILL      type is   fixed       3
          AWAR       type is   fixed       2
     Record name is WREC
          duplicates are not allowed for WCLASS,WTYPE
          WCLASS     type is   character   3
          WTYPE      type is   fixed       1
          WFF        type is   charactrer  3
          WRANGE     type is   fixed     . 5
          WFEUL      type is   fixed       4
          WLBS       type is   fixed       6
     Record name is  IDTEMP
          duplicates are not allowed for
                     IID, ICLASSS, ITYPE
          IID        type is   fixed       3
          ICLASS     type is   character   2
                     check is equal 'AC','SH','ARU'
          ITYPE      type is   fixed       1
     Record name is AW
          duplicates are not allowed for
                     WCLASS, WTYPE, WAMMO
          WCLASS     type is   character   3
          WTYPE      type is   fixed       1
          WAMMO      type is   character   1
     Record name is   TNUMBER
          TDAY       type is   fixed       3
          TPNUM      type is   fixed       4
```

Figure B.2 DBTG Record Schema Description

```
Set name is IR_IDTEMP
     Owner is IREC
     Order is  sorted by defined keys
     Member is  IDTEMP
     Insertion is automatic
     Retention is fixed
     Check is IID in IREC = IID in IDTEMP
         and  ICLASS in IREC = ICLASS in IDTEMP
         and  ITYPE  in IREC = ITYPE in IDTEMP
     Set selection is by value of
          IID, ITYPE, ICLASS
Set name is AW_AR
     Owner is AW
     Order is sorted by defined keys
     Member is AREC
     Insertion is automatic
     Retention is fixed
     Check is WAMMO in AW = ACAT in AREC
Set name is WR_AW
     Owner is WREC
     Order is sorted by defined keys
     Member is AW
     Check is WCLASS in WREC = WCLASS in AW
       and    WTYPE  in WREC = WTYPE  in AW
Set name is TOT_NUMBER
     Owner is PREC
     Order is last
     Member is  TNUMBER
     Insertion is manual
     Retention is optional
     Check is PDAY in PREC = PDAY in TNUMBER
     Set selection is by value of PDAY
Set name is PR_IDTEMP
     Owner is PREC
     Order is last
     Member is IDTEMP
     Check is PID in PREC = IID in IDTEMP
       and    PCLASS in PREC = ICLASS in IDTEMP
       and    PTYPE in PREC = ITYPE in IDTEMP
     Set selection  is by value of
               PID, PCLASS, PTYPE
Set name is IDTEMP_AW
     duplicates are not allowed for WCLASS,WTYPE
     Owner is IDTEMP
     Order is sorted by defined keys
     Member is AW
     Check is ICLASS in IDTEMP = WCLASS in AW
        and   ITYPE in IDTEMP = WTYPE in AW
```

Fig B.3 DBTG Schema Description for Intelligence

14. Fagin, Ronald. "A Normal Form for Relational Database That Is Based on Domains and Keys." In Transaction on Database Systems, Vol.6, No.3, September 1981

15. Codd, E.F. "Relational Database: A Practical Foundation for Productivity." In Communications of the ACM, Vol. 25, No. 2, February 1982.

LIST OF REFERENCES

1. Martin, J., Computer Data-Base Organization, Prentice-Hall Inc., Englewood Cliffs, N.J., 1977.

2. Ullman, J.D., Principle of Database System, Computer Science Press, Inc. , Rockville, Maryland, 1980.

3. Kronke, D., Database Processing, Science Research Associates Inc., Chicago, Tronto, 1983.

4. Date, C. J., An Introduction to Database Systems, Addition-Wesley Publishing Company INc., 1981.

5. Smith,D.C and Smith,J.M., "Conceptual Database Design" Tutorial on Software Design Techniques, 4th, 1983.

6. Hawryszkiewycz, I.T., Database Analysis and Design Science Research Associate, Inc. 1984.

7. Teorey.T.J. and Fry.J.P., Design of Database Structure Prentice-Hall, INC, Englewood Cliff N.J 07632. 1982.

8. Hammer, Michael, and McLeod, Dennis. "Database Description with SDM: A Semantic Database Model." Transaction on Database Systems, Vol.6, No 3, September 1981.

9. Berri, C, Bernstein, P. A., and Goodman, N. 1978. "A sophiscate's Introduction to Database Normalization Theory." Proceedings of the Fourth International Conference on Very Large Database, West Berlin, pp 113 - 124.

10. Zelkowitz, M.V., "Perspectives Software Engineering ", Computing Surveys, vol. 10, NO.2, June 1978.

11. Tsichristzis, D. C. and Lockovsky, F. H., Data Models, Prentice-Hall Inc., Englewood Cliffs, N.J., 1982.

12. E.F. Codd. "Recent Investigation into Relational Database Systems." Proc. IFIP Congress 1974. Also in Proc. ACM Pacific Conference, San Francisco, April 1975.

13. Codd, E. F. "A Relational Model of Data for Large Shared Database." In Communication of the ACM, Vol.8, No.6, June 1970

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Library, Code 0142 Naval Postgraduate School Monterey, Califonia 93943	2
2.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, Califonia 93943	2
3.	Professor S. H. Parry, Code 55Py Department of Operation Research Naval Postgraduate School Monterey, Califonia 93943	2
4.	Central Computer Center Army Headquarters, 140-01 Seoul, republic of Korea	2
5.	Jang, Jai Eun 6-512 Soldier APT Young-San Dong, Young-San Gu Seoul, Republic of Korea	8
6.	Chief of Staff Army Headquarters, 140-01 Seoul, Republic of Korea	1
7.	Department of Personnel Management Army Headquarters, 140-01 Seoul, Republic of Korea	1
8.	Division of Computer Management Department of Management Army Headquarters, 140-01 Seoul, Republic of Korea	2
9.	Division of Education Department of Personnel Management Army Headquarters, 140-01 Seoul, Republic of Korea	1

10.	Department of Intelligence Management Army Headquarters, 140-01 Seoul, Republic of Korea	1
11.	CDR Michael, J. Anderson Computer Technology Curriculum, Code 37 Naval Postgraduate School Monterey, California 93943	1
12.	Lee, Hee Young SMC #2726 Naval Postgraduate School Monterey, California 93943	1
12.	Lee, Jang Hun SMC #1366 Naval Postgraduate School Monterey, California 93943	1
13.	Song, Hwa Dal SMC #2748 Naval Postgraduate School Monterey, California 93943	1
13.	Choi, Kwang Jun SMC #2870 Naval Postgraduate School Monterey, California 93943	1
14.	Lee, Ju Kab SMC #2879 Naval Postgraduate School Monterey, California 93943	1
15.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2