DOCUMENT CONTROL AND RETRIEVAL SYSTEM FOR THE BRAZILIAN AIR FORCE

THESIS

Antonio F. Bernardo da Silva
Lt Col, Brazilian Air Force

AFIT/GCS/ENG/89J

Approved for public release; distribution unlimited.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/89J-2	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION AFIT/ENG
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Brazilian Air Force 8b. OFFICE SYMBOL (If applicable) BAF	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Document Control and Retrieval System for the Brazilian Air Force
UNCLASSIFIED

12. PERSONAL AUTHOR(S)
Antonio Fernando Bernardo da Silva, Lt Col, Brazilian Air Force

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1989 June	190

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Relational Database, Oracle RDBMS, Document Control, SQL, Document Retrieval, Thesaurus

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

This theses addresses the design and complete implementation of a document control system. The problem is defined, the requirements are specified, attributes are defined and the Entity-Relationship Model is used to describe the conceptual model. Entities and Relationship Sets are translated to tables and the database is implemented using the Oracle Relational Database Manager System, which features SQL*Forms, an interactive forms based system that accesses the Database using embedded SQL statements and triggers. The proposed system uses a Thesaurus to perform subject retrieving and provides the means to build and maintain the thesaurus. Keywords: Information retrieval; Air Force research; Brazil; Computer applications; relational database management system; Computer applications - Theses. (edc) ⊕

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL James Howatt, Major, USAF	22b. TELEPHONE (Include Area Code) (513) 255-6913 22c. OFFICE SYMBOL AFIT/ENG

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

AFIT/GCS/ENG/89J

DOCUMENT CONTROL AND RETRIEVAL SYSTEM FOR THE BRAZILIAN AIR FORCE

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Science

Antonio F. Bernardo da Silva, B.S.

Lt Col, Brazilian Air Force

June 1989

Approved for public release; distribution unlimited.

Preface

The purpose of this thesis effort was to design and implement a prototype of a document control and retrieval system which could be used as a standard system by Brazilian Air Force Organizations. The system is addressed to documents exchanged externally among Brazilian Air Force organizations and other government, civilian and private organizations.

The diversity of formats, standards, and kinds of documents make such a system a challenge. The belief that it can significantly improve decision making and assessment has motivated me to accomplish this task.

I would like to thank my thesis advisor, Major James Howatt, who has enriched this work with his significant comments and suggestions, and my committee members, Major Mark Roth and Prof. Henry Potoczny.

I wish to dedicate this work to my loving wife, Vera, for all the encouragement and support she has given me.

Table of Contents

List of Figures

List of Tables

DOCUMENT CONTROL AND RETRIEVAL SYSTEM FOR THE BRAZILIAN AIR FORCE

I. Introduction

Background

In this thesis the term documents refer to those printed reports sent or received by Brazilian Air Force (BAF) organizations, that constitute the Brazilian Air Force Official Correspondence. Because decisions, requests and reports are the essence of documents, the control and organization of a document collection has become as important as the decision making or/and assessment that depend on researching the collection. The use of digital computers profoundly affected the field of information retrieval by permitting the mechanization of routines and the scanning of large interdependent files. Because of the high costs involved, until few years ago most BAF organizations could not afford the computer resources necessary to run such systems. As a consequence of the electronics industry revolution that has been made microcomputers more powerful and cheaper, BAF organizations can now afford an increasing number of microcomputer systems, and some organizations can even buy such a system out of their own budget. It is already possible to take advantage of the new era and improve efficiency and speed in the classification, control, and retrieval of documents.

Problem Statement

The problem addressed in this thesis will be the development of a document control and retrieval system to manage Brazilian Air Force official correspondence. The system is intended to be used on microcomputers as standard software for most of BAF organizations.

Scope

This thesis presents the main concepts, the design and the implementation of a prototype of the system including the information retrieval subsystem that uses a thesaurus for searching by subject. The definition of the terms that will constitute the thesaurus is beyond the scope of this thesis, although the structure and rules to define them, as well as the tools to create and maintain the thesaurus, are given.

The procedures, terms, document numbering and database attributes comprise the standards for elaboration, archive and elimination of documents, as defined by the "Instrucao Sobre Correspondencia e Atos Oficiais do Ministerio da Aeronautica," ICAER (M.Aer., 1976), and by the "Regulamento para a Salvaguarda de Assuntos Sigilosos," RSAS (M.Aer., 1977).

As the official correspondence encompasses classified documents, it is assumed that the system will be operated by qualified personnel to which are granted the adequate access authorization. It is possible to limit the access to the system or parts of the system by using passwords. It is even possible to restrict the operation to a certain group of activities by using the facilities provided by the Data Base Management

System. These protection actions are beyond the scope of this thesis and thus, are not presented.

General Approach

The main steps followed in developing the system were:

a. To identify significant attributes of BAF documents.

b. To define the system requirements.

c. To review the literature on document control and information retrieval.

d. To review the literature on database design.

e. To review the literature on software engineering.

f. To choose the Database Management System (DBMS) for supporting the system.

g. To establish a software engineering environment, in hardware and software, to support the system development effort.

h. To design the relational database model.

i. To implement the prototype by modules.

j. To test each module individually and integrate it with the previous developed modules.

Sequence of Presentation

The ordering of the steps to come to a conclusion is not always the best ordering to explain that conclusion. The sequence of presentation differs a little from the ordering used to develop the system, but I believe that doing this the presentation becomes clearer. The second chapter, Requirements Specification, describes the desirable behavior of the system. In that chapter the system is decomposed into activities to perform the necessary functions. Chapter three reviews some of the theory that led to the design decisions made on the project, mainly on

the database design and on the choice of using a thesaurus. Chapter four presents the main aspects of the database design, which was modeled using the Entity-Relationship Database Model. The following chapter, number five, talks about the implementation of the model, which was done using the Oracle RDBMS for MDOS based PC microcomputers. In this chapter are presented some of the facilities offered by the Oracle RDBMS, like the 4th-generation tool to generate screen forms. Finally, chapter six concludes the work evaluating the proposed system and presenting recommendations for future research.

II. Requirements Specification

Introduction

Before presenting the requirements specification it is necessary to clarify some important concepts as which kind of documents the system is expect to deal with, which sectors in an organization are concerned with documentation, what is expected from a computerized system and which procedures of the manual system actually in use are important and must be preserved in the proposed system. Therefore, we present, as an introduction to the specification of the proposed system, a brief overview of the rules and procedures concerning document control and retrieval in the Brazilian Air Ministry, and some of the advantages of computerizing the manual system. All explanations are based in the official issues of ICAER (M.Aer., 1976) and RSAS (M.Aer., 1977). Following this overview we state the specification of the proposed system, which is divided in three major units, and identify the document attributes that will constitute each unit. Finally, the system is decomposed into modules to accomplish specific functions.

Document Control Overview

Document. The Air Ministry Official Correspondence, that is, the correspondence elaborated by or sent to organizations under the Air Ministry or Brazilian Air Force jurisdiction, may be divided into internal and external documents. Internal documents move internally among the sectors of the same organization. External documents are those exchanged among Air Ministry organizations or among these organizations

and organizations or persons that are not under the Air Ministry jurisdiction.

Protocol. An organization has two sectors that are directly concerned with documentation, the Protocol and the Archive sections. The Archive Section is introduced in the next paragraph. Protocol is the document register and delivery section, which is responsible also for giving to the users information about documents and document management. For each document that is sent or received, the Protocol Section keeps a "Ficha Protocolo" (FP), that is, a Protocol Card, which records the document movement. The FP, which is typewritten with three copies, for a total of 4 versions, contains the description of the document characteristics. Each copy is kept in a different file and they provide the retrieval of documents by:

1. Protocol Number
2. Originator
3. Name of a person related to the document
4. Subject (one line summary of the document)

An organization may have two Protocol Sections, one for public documents and the other for classified documents. Classified documents demand some supplementary procedures, as a document-received confirmation from the addressee, and sometimes use a more sophisticated subject retrieval, usually based on a list of subjects.

Archive. Archive is the document repository section. It is responsible for organizing the document collection in such a way that it can be easy consulted. There are three levels of Archive:

- of an internal Department or Division;
- of an Organization;
- of the Air Ministry.

In this effort we are mainly concerned with the Archive of an Organization. Table I shows which types of documents are kept in an organization.

Table I. Documents Kept in an Organization's Archive

Permanently:

- originals of Organization's Bulletin;
- documents with historical value;
- personnel records;
- documents related to Justice;
- documents related to Instruction (for schools, academies, institutes, etc);
- other documents, chosen by the commander.

For 10 years:

- documents related to permanent material.

For 5 years:

- documents related to finances;
- copies of documents sent to other organizations;
- documents related to discipline;
- received documents, definitely solved, which are not expected to be consulted again

Document Destruction. A commission composed by three members is designated yearly to select among the documents in the files those which should stay in the Archive, those to be sent to the Air Ministry Archive, or those to be destroyed. The decision is posted on the document control card. The procedure for classified documents is regulated by the RSAS (M.Aer., 1977b), but is essentially the same.

Microfilmed Documents. Microfilm process is used whenever possible. The documents to be translated to microfilm are those selected

to be kept permanently and for 10 years. Documents of historical value cannot be destroyed, even after they have been microfilmed. Microfilms should be organized to permit the same kind of retrieval that is done with documents in paper form.

What to Expect from the Automated System

One could argue that it is not necessary, nowadays, to discuss the benefits of automation because everybody knows that computers do it faster. I agree! The problem is why should we take the always limited resources we have, to computerize the document control system instead of some other not-automated-yet system. Is it important? How much time and manpower an automated system might save?

It is not possible to give precise numbers that represent the gain without having two real systems to contrast. The comparison depends on various parameters of each system being compared. In spite of not giving final numbers, we can present various aspects and facilities of both systems, to permit the reader come to his own conclusions. The remarks about manual and automated systems for document control are personal viewpoints derived from an experience of ten years working in tasks related to document management in the Brazilian Air Force.

Control. One of the most important advantages of the computerized system is the control over the documentation being processed. Once the criteria of stability was defined, that is, the average time to process kinds of documents, it is easier to detect instability and to take corrective actions. The manual system offers inadequate control. As an example, to find a document that has never reached the Archive Section we

have to follow the document track from its first destination to departments and sections through which it might eventually have passed. Another difficulty of the manual system is choosing which documents from the collection have to be destroyed. Deciding which documents are "definitely solved, and not expected to be consulted again" is a challenging task. Because of the inherent retrieval facility offered by computerized systems, it is possible to give a "validity" attribute to each document that makes easier a future decision about its destruction.

Standardization. A standard system offers the possibility of easy communication between distinct systems. It would be possible exchange documents in computer file form between organizations, taking advantage of the same kind of management, recording, and retrieval. This would represent a tremendous savings in personnel instruction, software development effort and time.

Manpower. When a manual document control system is automated the personnel staffing is not likely to be reduced for two reasons:

- usually the staff was insufficient to do all the tasks it was supposed to do. The automated system fills the gap.
- new facilities offered by the automated system and the completion of tasks which were not accomplished before absorb the manpower that would be saved.

Time. The following example gives an idea about time saving in an automated system. Correspondence arrival is not smoothly distributed, the regular official mail comes twice a day. Depending on the size of the organization, the Protocol Section is overloaded in these occasions. Maybe important information or urgent decisions have to wait for the protocol processing. In the manual system, for each document, the

Protocol has to typewrite data on a protocol card (four copies) and on a processing sheet. Suppose that the same data is to be entered in the computerized system using a keyboard and a terminal display, and that the processing sheet is printed by a printer connected to the computer. If we assume that:

a - the data about each document sums 180 characters on average

b - the operator types 180 characters per minute in average (typewriter or keyboard)

c - the time to change and adjust a protocol card in the typewriter is 30 seconds

d - the time to change and adjust a processing sheet in the typewriter is 30 seconds

e - the printer prints at 180 characters per second

f - the time to change and adjust a processing sheet in the printer is 5 seconds (it is automatic)

g - the operator does not make any mistakes entering the data

h - the Protocol Section receives, at the same time, 40 documents to be processed

i - the work is done by only one operator in either systems

Processing the 40 documents in the manual system would take 2 hours:

- Typing protocol cards: (h * a) / b = 40 * 180 / 180 = 40 minutes
- Typing processing sheets: (h * a) / b = 40 * 180 / 180 = 40 minutes
- Changing protocol cards on the typewriter: c * h = 0.5 * 40 = 20 minutes
- Changing processing sheets on the typewriter: d * h = 0.5 * 40 = 20 minutes

In the automated system the same task would take 0.73 hours:

- Entering the data on the keyboard: (h * a) / b = 40 * 180 / 180 = minutes
- Printing the processing sheets: (e + f) * h = 1sec + 5sec * 40 = 240 sec = 4 minutes

A very important and time consuming task was not taken into account. In the manual system we still have to insert each copy of the protocol cards in four different files, in alphabetical order. This totals 160 cards that have to be properly arranged. How much time does it takes? If it takes about 30 seconds to file each copy, this extra work will take 1 hour and 20 minutes. How many mistakes are committed in this task?

Reliability. Rigid and controlled structure is one of the most important characteristics of a computerized system. There is no threat of informal changes introduced in the system by users. There are means to validate entry data in order to reduce human errors. Once the entries are correct the data manipulation does not corrupt information stored in the database. Retrieval is less human dependent. We can browse the collection in numerous ways to assure that no documentation stays forgotten, inaccessible, lost in the stacks.

Resistance. Some resistance in accepting the new system is expected. Opposition usually is for unconscious reasons. Behind the antagonism there is a feeling of loss. Persons who used to be consulted about documents feel unimportant anymore because looking for a document was made easy. Delays and incorrect handling are soon detected and people feel they are being audited. Explanations about the system and its goals are the solution to the majority of resistance problems.

General Design Specification

According Pressman (Pressman, 1987:152), because specifications are the description of "what", rather than "how", they can take the form of mathematical functions: "given some set of input, produce a particular

set of outputs." Using this kind of description we present the desired system as follows:

Given a set of identifiers that denotes a document, the most frequent questions made about administrative documents are:

- Has the organization received this document?
- What is this document about?
- Where is this document?
- Which sections/departments have already processed this document?
- Who is currently analyzing this document?
- Which documents may I destroy?

Less frequent than the previous, but not less important, the following question arises:

- Which documents are related to this subject?

Figure 1 shows a typical administrative document routine for Brazilian Air Force organizations. The Protocol Section enters the document description in Protocol Cards and prepares a Processing Sheet (PS) that contains data about the document identification. The processing sheet is attached to the document. The Commander, or a person designated by him, decides which department or section the document will be sent to; this is explicitly placed on the PS. The Protocol Section sends the document to the referred department or section. If it is necessary, the former department sends the document to another department. After processing, the document is sent to the Archive Section. In the processing phase, occasionally, documents are joined to each other to form a single document. Briefly, these are the main steps involved in the administrative document routine.

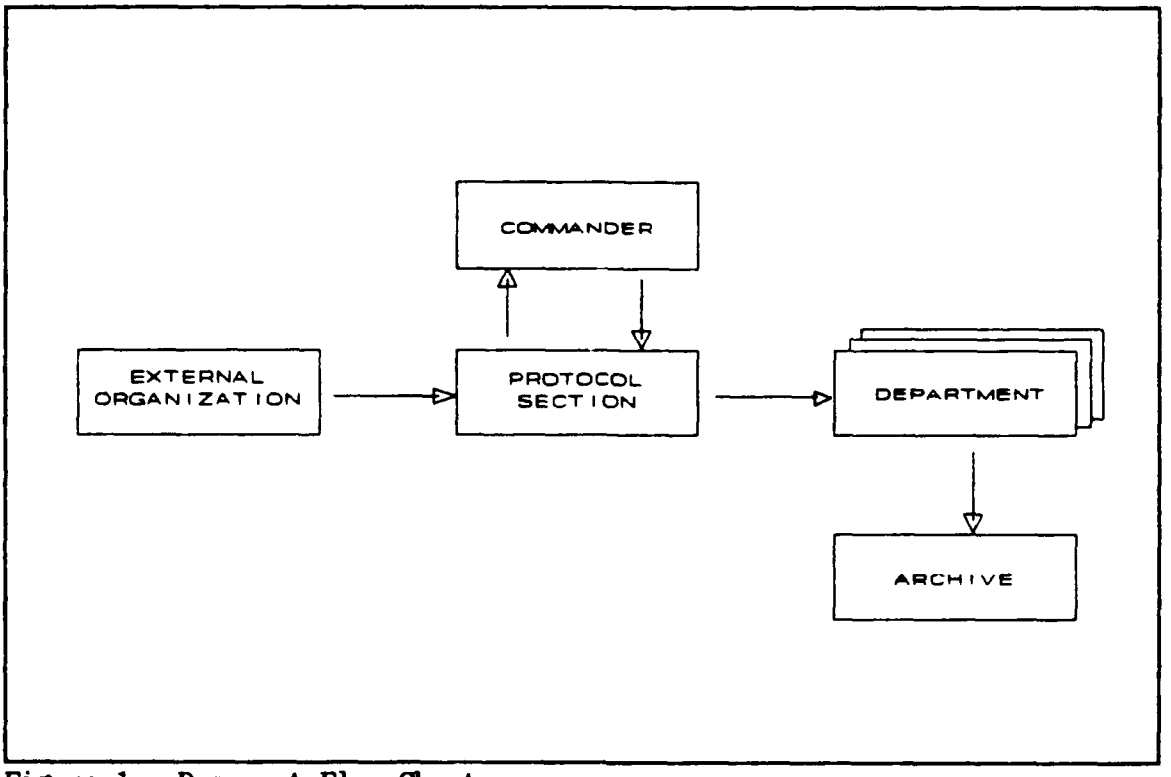


Figure 1. Document Flow Chart.

The Archive Section works as a library and keeps the whole document collection. Departments and sections borrow documents from the Archive. In the Archive Section, documents may stay in paper form or may be recorded on microfilm. Periodically, useless documents are selected to be destroyed.

To answer those questions about administrative documents, the proposed system has manual and computerized procedures that involve the registering, distribution, processing, retention, archive and destruction of documents, each one contributing to form a knowledge base about the document collection. This knowledge base is logically subdivided in three parts:

- Catalog Database
- Subject Database
- Control Database

We may thing about the system in an object-oriented manner where a document is the principal object. The logical division of the database reflects that we are mainly interested in three aspects of documents:

- The Catalog view, which gives us the identification of a document based on its characteristics; we can call these characteristics "header attributes".
- The Subject perspective, that classifies documents by their contents, that is, by the issues, themes, topics, or information addressed on them.
- Finally, the Control perception, which gives information about the document processing stage, the department or section that keeps the document, the person who is handling it, etc...

The two first databases are concerned with different kinds of document retrieval and the last one with the document control. In the following section the attributes that constitute the Catalog and Control Databases are presented. The attributes which form the Subject Database are introduced on the section about Thesaurus theory.

Document Attributes Identification

By document attributes we do not mean only those that are written in the header or body of a document, but also those that are used to identify or control the documentation. Some of them are found in the Processing Sheet, the paper attached to a received document by the Protocol Section, others may be written in a "despacho" (dispatch, process) cover, which is a kind of a document that is a join of other documents necessary to answer or clarify a same question or subject. Explanations will be given whenever necessary to clarify the meaning of an attribute.

Catalog Database. The catalog database is composed of all files that together store the document characteristics, generically called "header attributes." These attributes are listed in Table II and defined in the next paragraphs.

Table II. Header Attributes.

Protocol Number
Document Identification
Document Reference Number
Organization Name
Issuing Date
Title
Document Type
Classification
Reference
Annex

Protocol Number. The Protocol Number (PROT) is one of the major means of document identification in the Brazilian Air Ministry. This number is assigned to a document by the issuing organization and no further numbering is permitted, even when a receiving organization forwards the document to other organizations. It is composed of three parts: an Organization Code (P_CODE), a sequential number (P_NUMBER), and the two last digits of the current year (P_YEAR). The parts are separated by slashes. The P_CODE is formed by the superior command code (P_SC) and by the organization code (P_OC), which are separated by a bar (99-99).

Figure 2 shows an example of a protocol number. In a document, this number is always preceded by the initials "Proc". Air Ministry organizations which do not have the organization code (P_OC) substitute, on a temporary basis, the code by the organization initials. Any

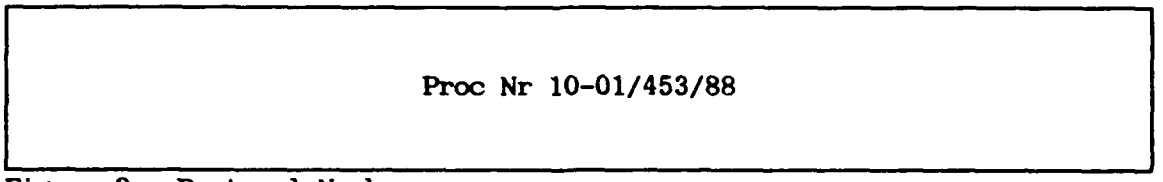


Figure 2. Protocol Number.

document received by Brazilian Air Ministry organizations that does not have the protocol number is numbered by the receiving organization. Each document has only one protocol number, from its issue to its final archive, even if it has been sent to several organizations in different years (M.Aer., 1976:7-1,7-2). Classified documents have an independent sequential ordering for each classification level.

Document Identification. The document identification is an alphanumeric identification given to a document by the issuing organization. This identification may be subdivided in a variable number of fields, each one with a different meaning. The segmentation and the meaning of each part vary among organizations. These variations make difficult the use of the document identification attribute as a searching key. Since all variations have a common point, - a sequential identification number, this number is extracted from the document identification and used isolated as a searching key. All other parts together constitute an identification complement to differentiate documents with a same identification number.

- Identification Number - IDNR
- Identification Complement - IDCOMPL

Figure 3 shows the placement of the identification number and complement on a document. Different documents, even from the same organization, might have duplicated IDNR. This typically occurs when each Department

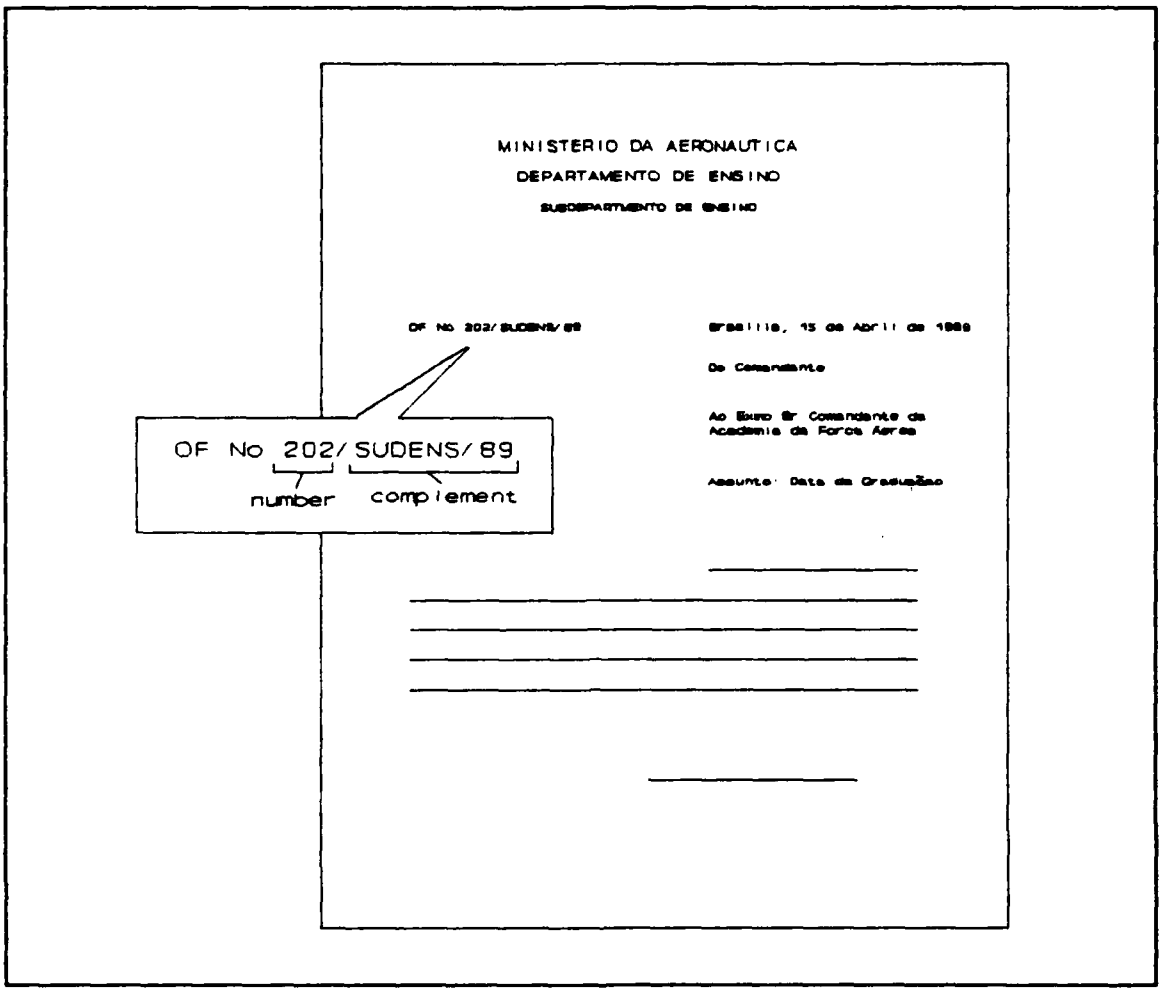


Figure 3. Identification Number and Complement.

in the same organization assigns its own number. Usually, the differentiation is made by the Department initials in the IDCOMPL.

Document Reference Number. The DRN is a composite number given to a document by the Archive Section. It is formed by:

- the year of registering
- the origin code ("1" received documents, "2" sent document)
- the sequential number under each of the previous numbers

The document collection is physically organized in the Archive Section by this number, making easy the physical document retrieval.

Organization Name. ORGNAME is the short name of the organization that has issued the document. There must be an unique name for each organization.

Issuing Date. ISSUEDT is the date when the document was issued. Issuing date and organization name together constitute an important key for an empiric retrieving, browsing a set of documents issued by a certain organization, during a given period of time.

Title. The TITLE of an administrative document usually is an outline of the Document subject. This title does not have the same meaning as a bibliographic or technical document title. It is not unusual to find several documents with the same title.

Document Type. The BAF deals roughly with 40 types of document - DOCTYPE. Most of them are sporadically used. Some of the most used documents are shown in Table III.

Classification. Concerning the distribution - CLASSIF, documents are considered public or classified. The public official issue that establishes special procedures applied to classified documents is the "Regulamento para a Salvaguarda de Assuntos Sigilosos" - RSAS (M.Aer., 1977). It divides classified documents in Reserved, Confidential, Secret and Ultra-Secret. As the details of each classification are not in the scope of this thesis, we will not discuss them here. The procedures for classified documents will be presented whenever it becomes necessary. The proposed system is intended to manage three of the classification levels:

P - public or unclassified
R - reserved
C - confidential

Table III. Some Types of Document used by the BAF.

Document Name	Translation
Apreciacao	Analysis
Despacho	Process, join of documents
Encaminhamento	Doc. listing and introduction of other documents
Estimativa	Estimating
Informacao	Information
Oficio	Officio
Relatorio	Report
Relatorio Especial	Special Report
Relatorio Periodico	Periodic Report
Requerimento	Request
Boletim	Organization's Bulletin

Reference. Some documents may refer (REFER) to other documents for various reasons. The organization may or may not have received the referenced document. The reference itself can be a Document Reference Number, a Protocol Number, a Document Identification or another alphanumeric string.

Annex. Documents may have a supplement attached to it. This supplement is called "Anexo" (ANNEX). This attribute describes in words the writing attached to the document.

Control Database. As the name indicates, the control database is related to the document control and to the actual phase of document processing. The attributes identified in this base are the following:

Validity. The interest about most administrative documents decreases with time. Some documents are of ephemeral importance and become of no interest very fast. As a consequence, unlike bibliographical techniques, we will be interested in destroying these

useless documents that occupy space in the Archive Section. When a document is cataloged in the system it automatically receives a validity date (VALIDITY), that consists in the date to which the document is considered of importance (valid). This automatic assignment is made for a five year period, based on the document Issuing Date. The document analyst, when processing the document, may request to change the validity date. The request is written on the processing sheet and processed by the Archive Section.

Document History. The document history expresses the processing for which a document has passed. It is composed of two parts: the document log and the actual holder. The document log is a set of attributes chronologically arranged in a table which reflects the document processing.

DATE	- Delivery date
TIME	- Delivery time
SEND_DEP	- Sending department or section Initials
REC_DEP	- Receiving department or section Initials

The holder information is composed of the holding department code and the holding person name as follows:

DEPCODE	- Department code
NAME	- Name of who is processing the document

Too much control tends to slow down the document processing speed. From a centralized control overview it is enough to determine in which Department the document actually is. Therefore, the name of the holder is an accurate control to be optionally used by a Department as an internal control, whenever the department has its own computer terminal.

The document history is updated when the document is sent from one Department to another one. The information for this update is extracted from a transference receipt.

Document Status. The document status represents the document situation related to the following conditions:

JOINSTS - Join Status

- J - Joined. The document was joined to another one. In this situation it does not have its own document history. The history of a joined document is the same as the history of the principal document, that is, the document to which it was joined.
- M - Main document. Other document(s) has(ve) been joined to this document.
- I - Independent. The document has its own document history.

PROCSTS - Processing Phase Status

- P - Processing. The document is being processed. It is automatically given by the system when the document is entered.
- W - Waiting. Unpredictable delay on the document solution or unsolved document.
- S - Solved.

FORMSTS - Document Form Status

- P - Paper.
- M - Microfilm.
- F - Computer File.
- D - Destroyed. The document has been destroyed.

HISTSTS - Document History Status

- A - Alteration. An anomaly or inconsistency was detected in the document history.
- N - Normal.

ARCHSTS - Responsible Archive Section Status

- C - Classified Archive.
- U - Unclassified Archive.

To know whether a document is filed in the classified archive or in the unclassified archive it is not always enough to know the document classification level. Although a classified document will never be filed

in the unclassified archive, the inverse may happen. If an unclassified document is joined to a classified one the whole document becomes classified. A controversy arises whether the unclassified part should or not be officially declared classified. The Archive Status solves this uncertainty by giving the exact direction.

Microfilm Indexing. When the document has been microfilmed, the kind of film and frame identification are recorded. They might have more than one microfilm record related to each document because frames are not in sequence or are in distinct films. The database attributes are as follows:

F_TYPE - Type of film
F_NO - Film number
FIRST - Initial frame number
LAST - Last frame number

Join Indexing. When documents have been joined, which is indicated by the join status, there is a list that relates the main document to the document(s) that has(have) been joined to it. On this list documents are referred as:

MAINDOC - the main document
JOINED - the joined document(s)

Verifier. VERIFIER is a control attribute created to protect against wrong document updates. It is used in update screens that do not display the catalog data, which makes difficult the document identification. Its computation was derived from a Division Hash Function where we consider the verifier as the resulting address for a DRN key. The original hash function was slightly modified by taking the DRN to the second power, to provide no consecutive verifiers for consecutive DRNs. Figure 4 shows the original function, the Verifier

Hash Function: F(X) = X mod M

Verifier Function: Ver(DRN) = mod(DRN^2,97)

RECEIVE DRN	VERIFIER	SEND DRN	VERIFIER
89100001	8	89200001	61
89100002	50	89200002	89
89100003	94	89200003	22
89100004	43	89200004	54
89100005	91	89200005	88
89100006	44	89200006	27
89100007	96	89200007	65
89100008	53	89200008	8
89100009	12	89200009	50
89100010	70	89200010	94

Figure 4. Verifier Function.

Function, and some DRNs with corresponding Verifier values. "M" was chosen as 97, which is the highest two digit prime number, giving verifiers in the range 0 to 96. The choice of a prime number is suggested by Horowitz (Horowitz, 1987:456,457). The Verifier is printed on the Processing Sheet.

Modular Partition

There are various analysis methodologies that enable an analyst to apply fundamental analysis principles. According Pressman (Pressman, 1987:164), all methods have the common characteristics of permitting a function representation, definition of interfaces, mechanisms for problem partitioning, support for abstraction, and representation of physical and logical views. Structured Analysis and Design Technique, SADT (registered trademark of SofTech, Inc.) is a comprehensive methodology for doing functional analysis and system design. It is beyond the scope of this

work to detail this technique but its fairly strict syntax permits a good understanding of the system decomposition without deep explanations. This modeling technique encompasses a series of hierarchically related function diagrams. It is used in this work to capture the system decomposition from the user viewpoint and to give us a first insight of modules interaction.

The Environment Level Diagram. The environment level diagram shown in Figure 5 is the system highest level diagram. It shows the interaction of the system with the world. On the left side there are the inputs to the system. I1, Keyboard Input, represents all information entered via the terminal keyboard. This information might be data from the document header, parameters for a document search, or any other kind of information needed to update the database files or to run the system programs. I2 represents all the database files used by the system to storage information, either in a temporary or permanent basis. C1, User Commands, on the upper side of the figure, defines to the system what to do with the entered input. User commands are the menu choices, function keys, and any other means of choosing an action to be taken by the system. These actions and the input elaboration produce the outputs, which are interpreted on the right side as CRT Displays - O1, Printed Reports - O2, and Modified Database Files - O3. The modified database files are reused as input, which is represented on the diagram by the cycle from the output to the input.

The Main Modules. Figure 6 shows the A0 level SADT diagram corresponding to the environment diagram of Figure 5. Although meaningful, the logical databases in which the system is divided are very

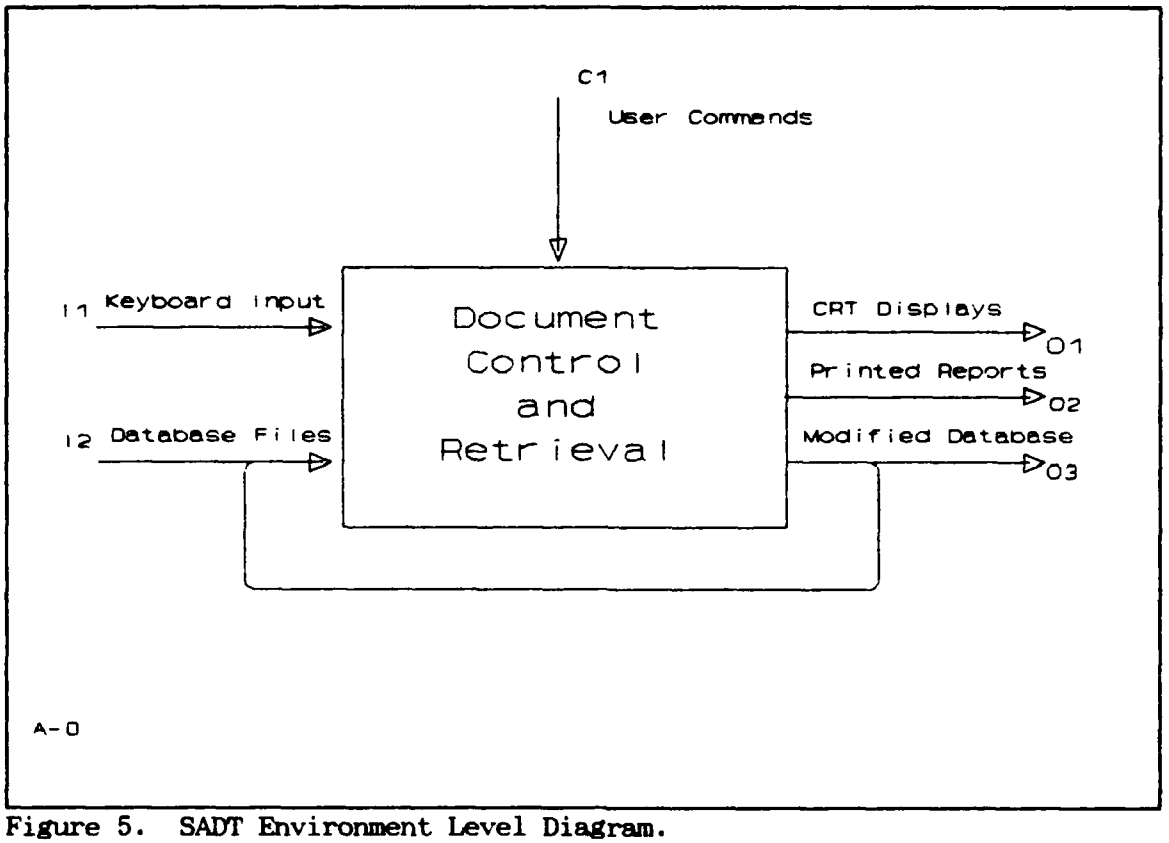


Figure 5. SADT Environment Level Diagram.

ample. Each one embraces several attributes that are not available at the same time. We need modular partitions that depict the available operations for the object document. These operations will add and modify attribute values in each relational database. They are grouped in the four main modules shown in Figure 6 as follows:

1. Register Document
2. Update Database
3. Consult Database
4. Maintain Thesaurus

Although modularity has been accepted as an important software attribute and an extension of the "divide and conquer" method, we must strive for a few design heuristics. At this level, these heuristics are maximum cohesion and minimum coupling between modules to improve modular

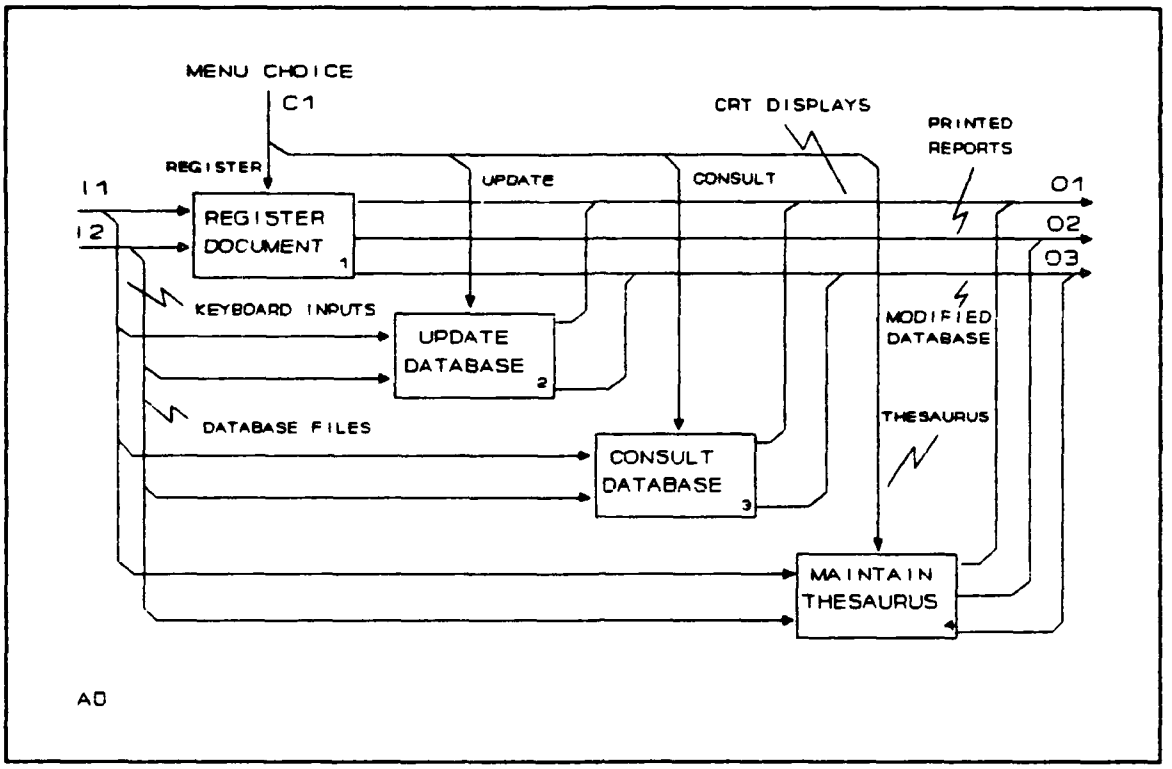


Figure 6. SADT A0 Level Diagram.

independence and clarity of design. The variety of operations encompassed by the system constrain our first level decomposition (A0) to a logic and temporal cohesion, which will permit later, the desired functional cohesion. On the other hand, they present the desired loose coupling, with no direct coupling between modules.

Register Document. Figure 7 shows the decomposition of the Register Document module. Note that this module is numbered 1 in Figure 6, and that its decomposition, Figure 7, is named A1 on the left bottom corner. This is the guiding line to relate a parent module to its decomposition. The document processing routine begins with entering data about just received documents. This data is extracted from the document header. The system gives to each document an unique Document Reference

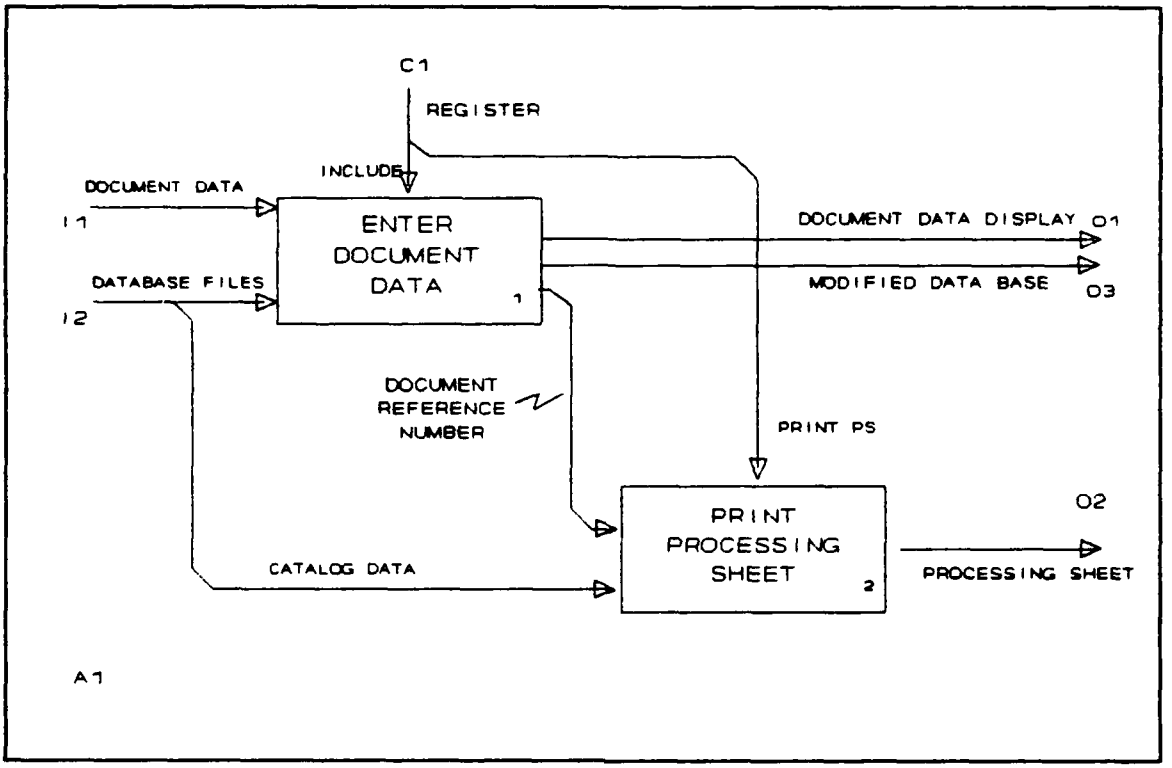


Figure 7. Register Document Diagram.

Number - DRN. The Document Reference Number that comes from module A11, Enter Document Data, to module A12, Print Processing Sheet, points to the last document entered, as a reminder to the user when printing the processing sheet - PS. Although these modules are independent of each other, the typical sequence is first entering all received documents, then printing the corresponding processing sheets.

Update Database. Figure 8 displays the decomposition of the Update Database Module. The decomposition presents functional and procedural cohesion. Module A21, Generic Update is intended for generic situations where some erroneous data was introduced in the catalog database and the correction using some of the other modules is not

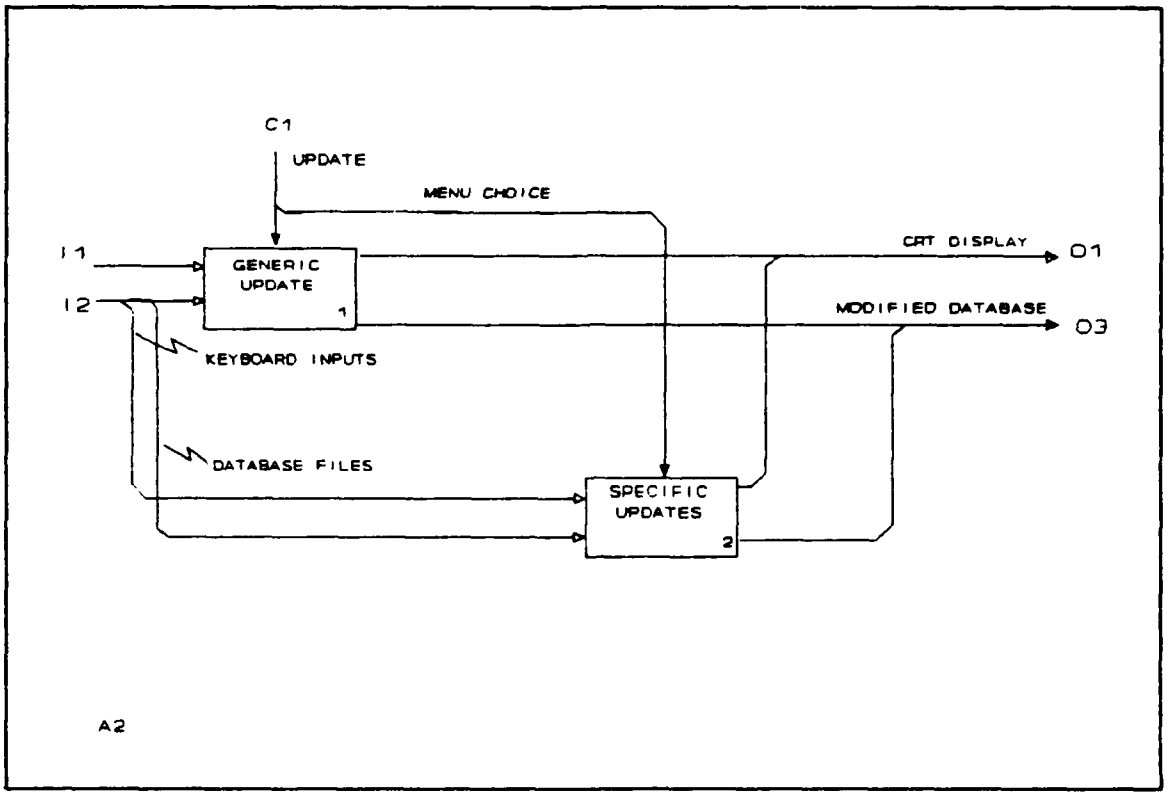


Figure 8. Update Database Diagram.

possible or convenient. Thus, this module is mainly related to the Protocol or Archive Sections. The Module A22, Specific Updates, encompasses several update operations, which are described in the following section.

Specific Updates. This module is formed of various lower level modules.

- Change Validity
- Process Document Destruction
- Join Documents
- Transfer Custody
- Update Microfilmed Data
- Subject Indexing

Although they all modify the control database they do so on distinct occasions and for different purposes. Therefore, they are separated into individual units. There is no SADT Diagram describing these modules but they are shown in Figure 11, as part of the system structure.

- Change Validity is used by the Archive Section when an analyst requests a change to the validity that was automatically given to the document.

- Process Document Destruction removes some data from the files (as document history, for instance), and references the destruction authorization document.

- Join Documents relates a document (main document) to other ones (secondary documents) when these documents are joined to constitute a single document. This update is to be made by a department or by the Archive Section. This module also separates Documents, giving them again their own individuality.

- The Transfer Custody module updates the document history. Each document transference is made based on the Document Transference Receipt (see Figure Figure 23), that constitutes the sender's transference proof. The details of this receipt are introduced in the implementation chapter.

- Update Microfilm Data adds to the database information about microfilmed documents.

- The Subject Indexing module creates and modifies relationships among documents and thesaurus terms.

Consult Database. Figure 9 shows the Consult Database Mode. The module A31, Browse Database, permits an overview of several documents at the same time. It is further decomposed into three modules, Browse by Holder, Browse by Header, and Browse by Subject. They are specially useful when the searcher is not sure about the document he is looking for or the adequate keys to use in a query. The module Browse by Holder permits a Department to control its own document processing. As was stated before, the name of the person who is actually analyzing a document is not compulsory information. Although from a centralized

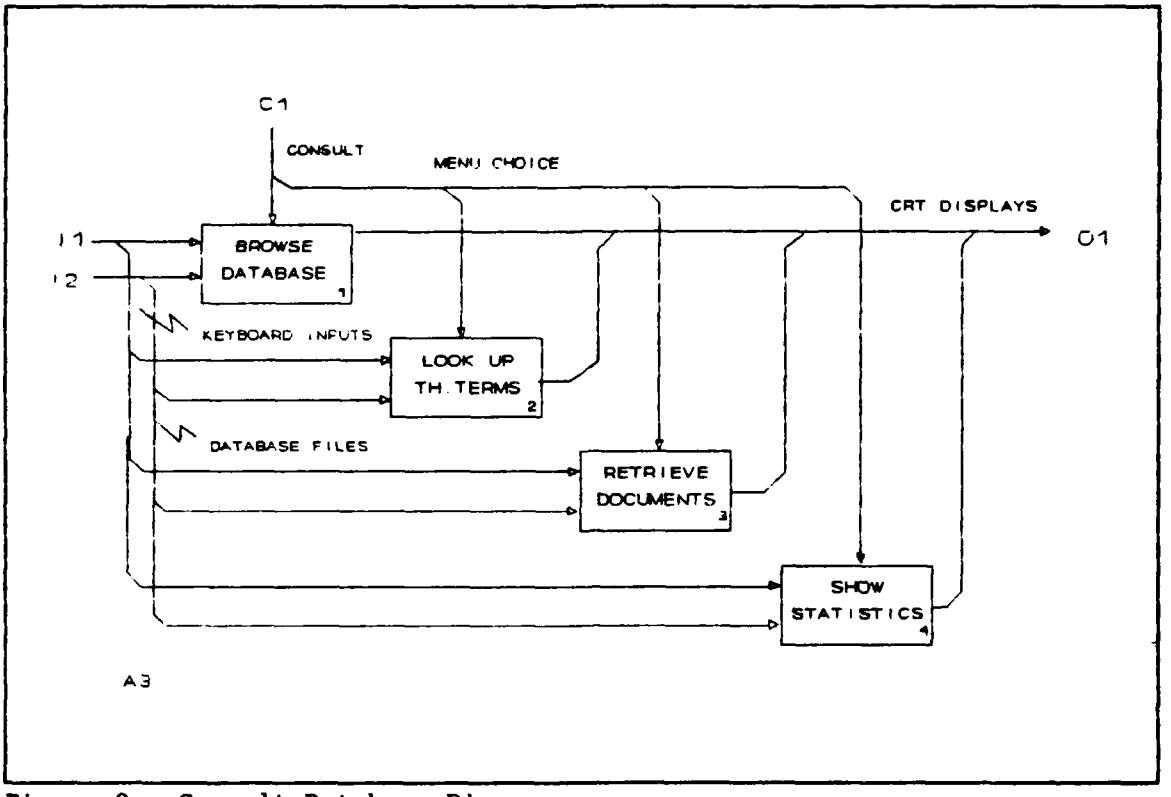


Figure 9. Consult Database Diagram.

viewpoint is enough to know the Department responsible for a document, these modules can and should be freely used by a Department as its internal control over the documents that are being processed and to identify the person who is processing each one of them.

Look Up Thesaurus Terms is a feature to search for a thesaurus term that approximates some desired concept. It permits the user to navigate through the thesaurus tree.

The Retrieve Documents module permits a document search by any document attribute and gives a complete information about the documents retrieved. The result of a query is a list of the documents that satisfy the request. These documents may be seen one at a time on the screen.

Maintain Thesaurus. Figure 10 shows the decomposition of the Maintain Thesaurus Module. Observe that the thesaurus maintenance is quite independent of the other main modules. As a standard system used by various organizations, we are interested in keeping identical thesaurus terms. This is easier accomplished by having an unified thesaurus maintenance. For this reason only the organization responsible for the thesaurus maintenance needs the complete Thesaurus Maintenance Module. This organization would concentrate the requests for changes and distribute upgraded versions of the thesaurus terms.

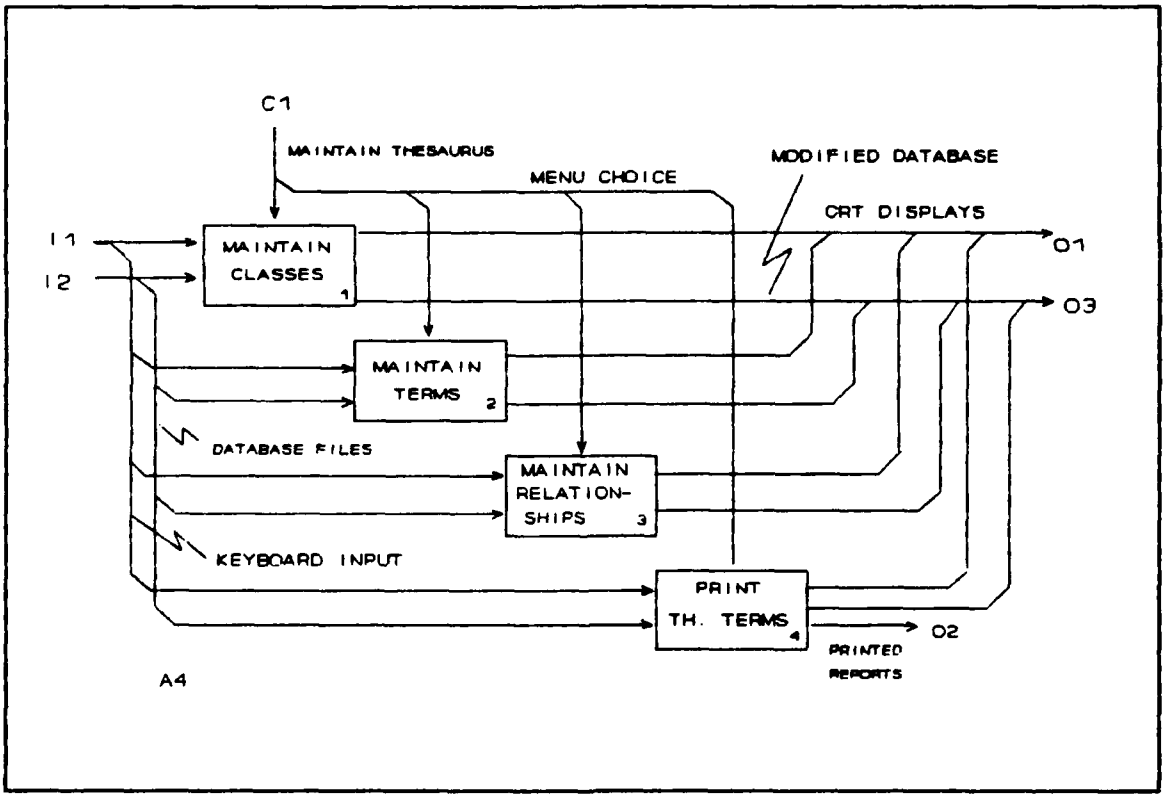


Figure 10. Maintain Thesaurus Diagram.

The System Structure

The system structure chosen, presented in Figure 11, closely resembles the decomposition already made. The necessity for defining the interfaces between modules is attenuated by using a Database Management System (DBMS), which consists of a collection of interrelated data and programs to access that data. Database systems are designed to manage large bodies of information. Therefore, they already provide a storage structure and mechanism for the manipulation of information, including provisions for the safety of the information stored. On the other hand the data structures become more important. The data structures are expressed as the database model that permits the DBMS to access the necessary information in the files. The database model presented in the next chapter.

To present the complete structure of the system we have anticipated the modules that constitute the thesaurus maintenance, although reserving for the next chapter the explanations about the thesaurus theory.

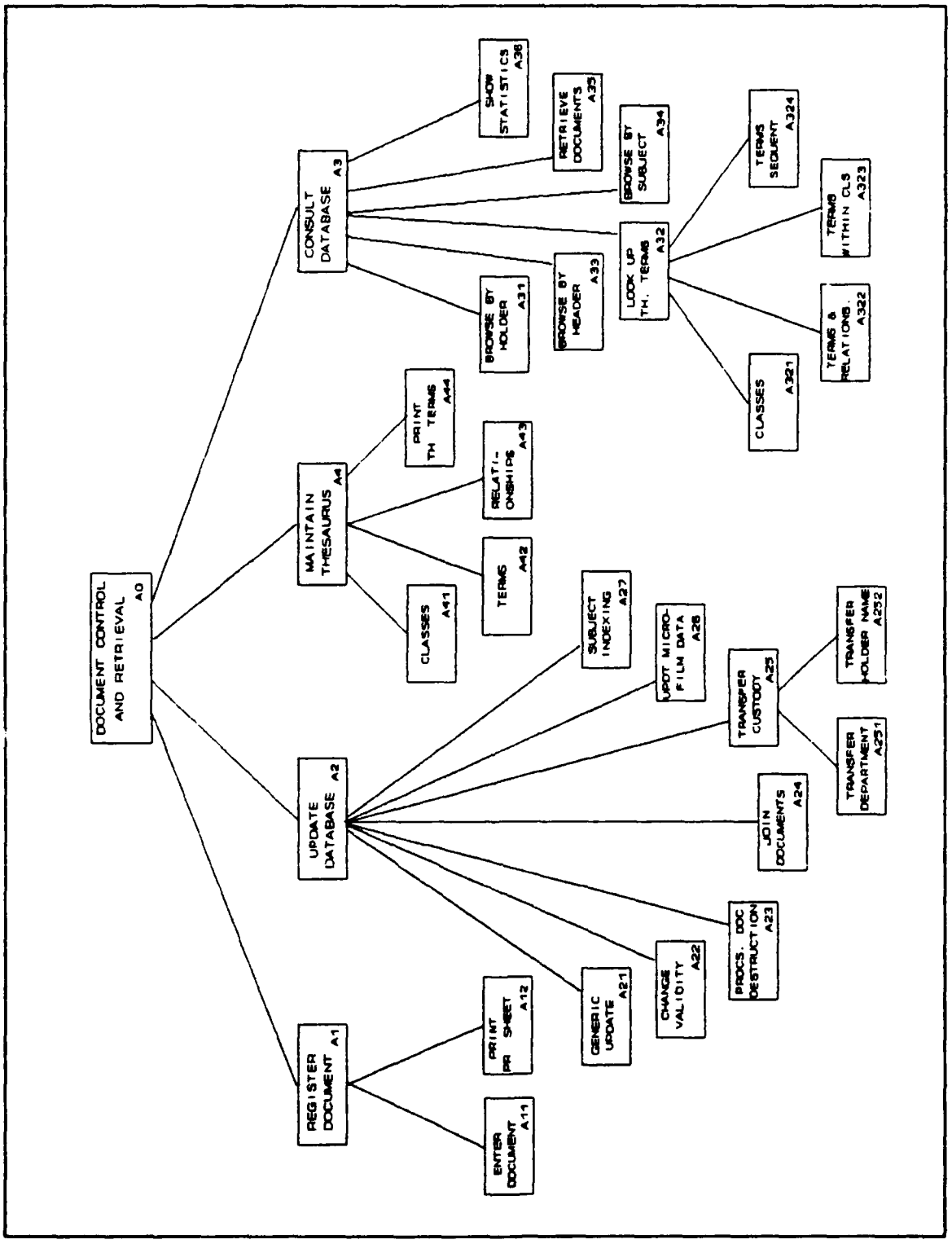


Figure 11. System Structure Diagram.

III. Theory

Thesaurus

The thesaurus is the heart of the subject database. To better understand the distinct roles that the subject database and the catalog database play we may divide the relevant information for retrieving documents from the collection into two units:

1. A set of data which defines a specific document, that is, the document identification or physical description.
2. A subjective set of terms which describes the document contents, that is, the information the document transmits.

The first one involves a process called cataloging and the second one subject indexing. The subject indexing process has two steps:

- the conceptual analysis of a document.
- the translation of the conceptual analysis into a particular vocabulary.

The translation of the conceptual analysis into a particular vocabulary employs a controlled vocabulary, that is, a limited set of terms used by both the indexer and the searcher to represent the document contents. The vocabulary might vary from a simple list of subject headings, keywords or phrases, to a thesaurus. The proposed system uses a Thesaurus that is the most used approach in information retrieval applications (Jackson, 1971:1). In the thesaurus approach, a document is regarded as a list of terms. A decision to retrieve or not retrieve is made by determining whether the list of terms which represents the document matches the terms of the request. The whole vocabulary is logically subdivided into individual vocabularies or subsets, each one

pertaining to different subject areas called Classes. The vocabulary is structured to show relationships between terms. Strict rules are established to define permissible structuring. The relationship between two terms may be expressed by a three element tuple where the first element is the main term - MAINTR, the third element is the secondary term - SECTR, and the second element is the two-letter code - REL, which expresses the relationship that relates the main term to the secondary term. The relationships implemented in the subject database are:

Hierarchical Relationship.

- Broader Term (BT)
- Narrower Term (NT)

The hierarchical relationship is represented by the codes BT and NT. The first one means broader term and the second one narrower term. As an example, the hierarchical relationships that are depicted on Figure 12 would be expressed as:

```
<POLICY              ,NT,POLITICAL DOCTRINES>
<POLICY              ,NT,EXTERNAL POLICY    >
<POLICY              ,NT,INTERNAL POLICY    >
<POLICY              ,NT,SECTORAL POLICY    >
<INTERNAL POLICY,NT,FEDERAL POLICY          >
<INTERNAL POLICY,NT,STATE POLICY            >
<SECTORAL POLICY,NT,AGRICULTURAL POLICY>
<SECTORAL POLICY,NT,TRANSPORT POLICY        >
<SECTORAL POLICY,NT,ENERGY POLICY           >
```

Those tuples only describe half of the hierarchical relationships that are depicted on Figure 12. Two rows are necessary to completely express each single hierarchical relationship between two terms, that is, we have to create reciprocal entries. The reciprocal is obtained by switching the main and secondary term and replacing the relationship code by its inverse, that is, NT by BT (or BT by NT).

```
<POLITICAL DOCTRINES,BT,POLICY              >
```

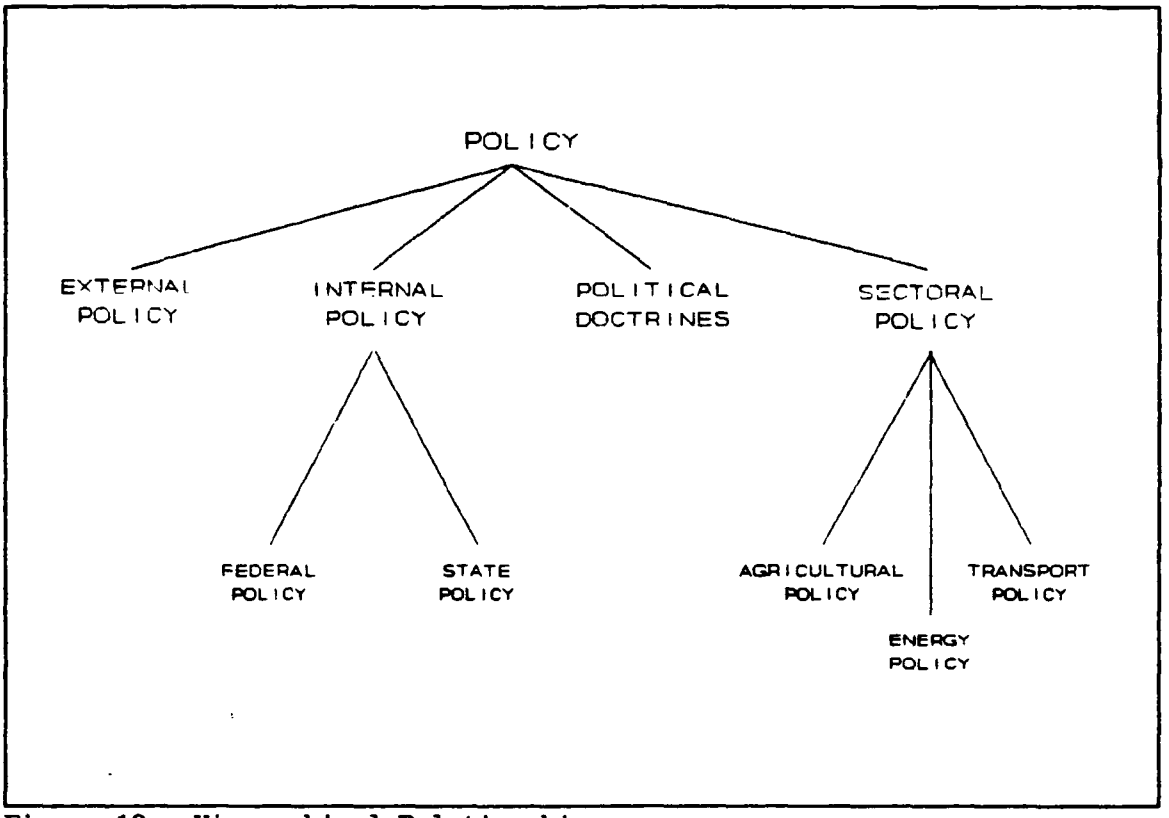


Figure 12. Hierarchical Relationship.

```
<EXTERNAL POLICY     ,BT,POLICY          >
<INTERNAL POLICY     ,BT,POLICY          >
<SECTORAL POLICY     ,BT,POLICY          >
<FEDERAL POLICY      ,BT,INTERNAL POLICY>
<STATE POLICY        ,BT,INTERNAL POLICY>
<AGRICULTURAL POLICY,BT,SECTORAL POLICY>
<TRANSPORT POLICY    ,BT,SECTORAL POLICY>
<ENERGY POLICY       ,BT,SECTORAL POLICY>
```

When retrieving documents, the replacement of a term by a broader term has the effect of increasing the number of documents retrieved. In the other hand, we might retrieve documents which are not relevant. The replacement of a term by a narrower term has the reverse effect. A term cannot have more than one broader term relationship.

Associative Relationship.

. Related Term (RT)

The associative relationship is represented by the code RT which means related term. It is used to approximate similar terms, which are alphabetically or hierarchically separated, and related terms, which are frequently associated during retrieval operations. The necessity of using such relationship may arise from relating:

- a thing and its application
- an effect and a cause
- a thing and property strongly associated with it
- a raw material and a product
- two complementary activities
- an activity and an agent of that activity
- an activity an a product of that activity
- a thing and its parts

[Lancaster, 1986:46-47]

A simple example of this kind of relation is the association of computers with keyboards and printers. Since the hierarchical relationship would be ambiguous, this relationship is expressed as:

<COMPUTERS,RT,KEYBOARDS>
<COMPUTERS,RT,PRINTERS >

<KEYBOARDS,RT,COMPUTERS>
<PRINTERS ,RT,COMPUTERS>

Note that the associative relationship also requires reciprocal entries. The reciprocal is obtained by switching the main term and secondary term of the original tuple.

Synonymy Relationship.

.Use (US)
.Used For (UF)

The synonymy relationship is represented by the codes US an UF. The first one means "use" and the second one means "used for." This

relationship assigns a single term to represent each unique concept. Other terms considered to be synonymous or near-synonymous indicate the elected term. In turn, the single term indicates the other terms it represents. The following entries would represent a synonymy relationship among priest, pastor, reverend, rabbi, and clergyman:

```
<PRIEST    ,UF,PASTOR    >
<PRIEST    ,UF,REVEREND  >
<PRIEST    ,UF,RABBI     >
<PRIEST    ,UF,CLERGYMAN>

<PASTOR    ,US,PRIEST    >
<REVEREND  ,US,PRIEST    >
<RABBI     ,US,PRIEST    >
<CLERGYMAN,US,PRIEST     >
```

A term that has a "use" relationship cannot have any other relationship. Since we are indicating another term to the user, all further relationships should be applied to the indicated term.

The relationships between terms are the thesaurus essence, but other attributes are also necessary to implement the vocabulary. The following paragraphs formally identify all attributes for the Subject Database.

Term Description - TERM. The term description expresses the desired concept and might be formed by one or more words. Each Term Description must be unique. Suppose we have a class named cities and another class named states. If we have a city and a state with the same name we differentiate them by adding its class (between parentheses) just after the word. For instance, suppose that cities pertain to class 05 and states pertain to class 06. To enter SAO PAULO city and SAO PAULO state we would enter these terms as: SAO PAULO(05) and SAO PAULO(06).

Class - CLASS. A class logically subdivides the whole vocabulary into individual vocabularies or subsets, each one pertaining to a different subject area. A document may be indexed under terms pertaining to a various classes. The searcher is able to retrieve a document not only by the precise term but also by the combination of terms pertaining to distinct classes. Moreover, the searcher is able to select all documents that we indexed by terms hierarchically subordinated to a chosen term. A class name is also a term in the thesaurus and all terms that have hierarchical relationships are descendants of their classes. Classes are intended for two main purposes:

- to facilitate the choice of terms. Dividing the all vocabulary into logically related smaller sets gives the searcher an intuitive notion of where to look for a desired term;
- to permit the system to use a set of terms as consistency constraints. For instance, organization names might be a class used to verify the database entries for document senders.

Class Code - CLASSCD. The class code was created to reduce disk space needs. As the class is repeatedly used, the internal use of such a code to represent the class avoids wasted space.

Main Term-MAINTR. A main term is a thesaurus term that constitutes the domain term of a relationship between terms.

Secondary Term - SECTR. A secondary term is a thesaurus term that constitutes the codomain of a relationship between terms.

Relationship Between terms - REL. The relationship between terms is a two-letter code that represents the relation between two terms in the thesaurus.

BT Broader Term
NT Narrower Term
RT Related Term

US Use
UF Used For

Scope Notes - NOTE. Scope notes are used to define or clarify ambiguous terms, to provide complete spelling of truncated terms, or to provide historical data on changes in usage for a given term. It is a guide to how a particular term should or should not be used. For instance, suppose we have the term "COMGAR" that pertains to the class "Organization". Its Scope Note can be the complete name of the organization: "Comando Geral do Ar."

Figure 13 illustrates possible relationships of a main term. Note that a term may have at most one broader term, while having several

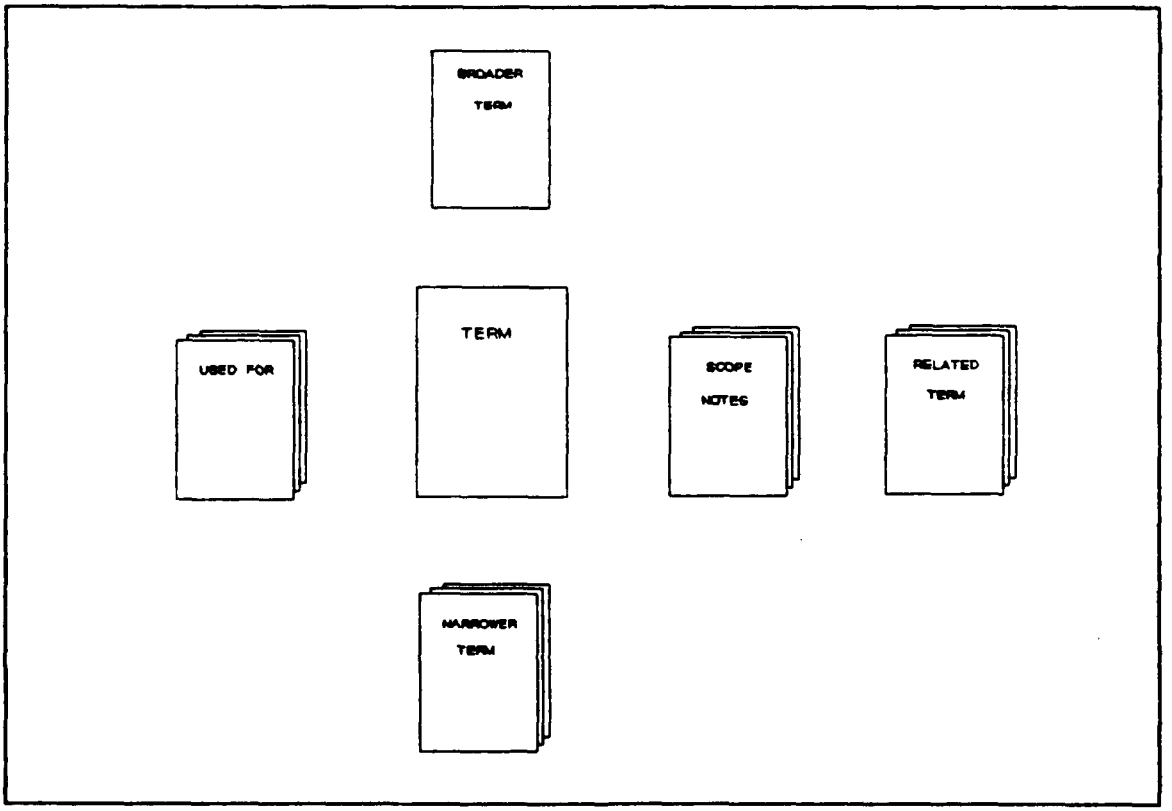


Figure 13. Main Term and Possible Relationships.

narrower terms. The same term cannot have "use" and "used for" relationships simultaneously. A term that has a "use" relationship cannot have any other relationship.

Database

In the following sections we explain some fundamental concepts about data models, data manipulation languages, and database management systems used in constructing the proposed system.

The Entity-Relationship Data Model. The Entity-Relationship Data Model (E-R) has been used for performing logical database design for relational systems. It is used in describing data at the conceptual level because it is semantically rich. The E-R model perceives the world as entities and relationships. The model provides for a high degree of data independence and is based on set and relation theories. The logical structure of the entity-relationship data model is graphically expressed as follows:

Rectangles. Represent entity sets. An entity is an object that exists and is distinguishable from other objects. A double line rectangle represents a weak entity. A weak entity does not have a candidate key and its existence depends on the existence of a regular entity.

Ellipses. Indicate attributes of an entity. An entity may be represented by a set of attributes. A star inside a ellipse denotes a candidate key for the set of attributes.

Diamonds. Denote relationships among entity sets.

Lines. Link attributes to entity sets and entity sets to relationships (Korth and Silberschatz, 1986:6-7).

Data Independence. There are two levels of data independence, the physical and the logical levels. This means that the programs to access the database do not have to be rewritten after a physical or logical modification on the database. A physical modification is a change in how the data are stored. A logical modification is a change in what data are actually stored in the database and the relationship that exist among data.

Data Integrity. This means that the data in the database have to satisfy some integrity constraints that are specified explicitly by the database administrator. As an example, suppose the constraint of a term name which has to be unique. It should be impossible to enter a new term with the same name as an existing term. A particular case of data integrity which the DBMS must ensure is known as "data consistency". It consists of keeping contradictory information from the database. An example is a same term with "use" and "used for" relationships simultaneously.

Security Enforcement. The data stored in the database needs to be protected from unauthorized access, intentional destruction, and malicious alteration. While "integrity" refers to accidental loss of consistency, security usually is related to protection against deliberated misuse. A DBMS usually provides security by using passwords which authorizes the access to only portions of the database and to perform limited operations.

Backup and Recovery. A computer failure due to mechanical or electrical problems may result in data loss. The DBMS has to offer means to detect such occurrences and to restore the database to a state that existed prior to the failure.

Concurrence Control. When several users may update the database at the same time there is a threat of data inconsistency. The DBMS has to control these interactions in a such a way that no harm can result from these operations.

Data Manipulation Language - DML. Commands are entered by the user and processed by the database management system. These commands, also called statements, and the rules that constraint their use constitutes the data manipulation language. Every DBMS possesses its own database language. Although there are differences among database languages they are divided into two basic groups: "procedural" and "nonprocedural." A procedural DML requires the user to specify which data is needed and how to get it, that is, the user provides the sequence of operations that generates the desired answer. A nonprocedural DML just requires an user to specify what data is needed.

A relational DBMS manipulates the data according the concepts and ideas of that relational data model described earlier (see page 41). They use manipulation (query) languages generically known as "relational database languages", which are classified as nonprocedural languages. SQL, Structured Query Language, is one of the best known relational database languages. SQL can be used in two modes: interactive, and embedded. In interactive SQL, statements are entered at a terminal or microcomputer and immediately processed or interpreted. In the embedded

SQL, statements are embedded in a program written in another language (procedural). These statements are not immediately executed. They are processed only when the host program is run.

The basic SQL expression consists of three clauses: select, from, and where. The "select" clause lists the desired attributes or columns of a table. The "from" clause lists the relations or tables that will be scanned in the execution of the expression. Finally, the "where" clause establishes certain conditions involving attributes that have to be met.

ORACLE Relational Database Management System. The Oracle RDBMS was the database management system chosen to implement the proposed system. It is a relational database designed to run on a microcomputer equivalent to the IBM PC/AT, which may be attached to a computer network. The system is compatible with S/ and DB2, which are IBM database systems that run on large IBM computers. According the Oracle manual (Oracle, 1987:3) Oracle runs on many different mainframe computers, minicomputers and microcomputers, being available on over 30 operating stems and 80 hardware platforms. Its Query language, SQL*Plus, offers a rich collection of features. The system offers in addition SQL*Forms, an application development tool also integrated to the system that permits a quick development of forms-based applications for entering, querying, updating and deleting data. Forms are specified using menus and a screen painter. Instructions and embedded SQL statements are combined to generate the application. More explanations about the system will be given in the chapter describing the implementation.

IV. Data Base Design

Introduction

According the "Webster's New Word Dictionary" (Simon and Schuster, 1984:360), the definition of database is as follows:

> "A large collection of data in a computer organized so that it can be expanded, updated, and retrieved rapidly for various uses.

This collection of data has to be organized in some way. Moreover, we need a set of programs to manage the data files in order to expand, to update, and to retrieve information. We used the Entity-Relationship data model for performing the logical database design. The E-R model was introduced by Peter Chen in 1976. Most of the concepts presented in this section are borrowed from him (Chen, 1976).

The logical view of data is an important issue. The E-R model describes the world in a natural and intuitive way that consists of entities and relationships. In this work the E-R model is used at two levels: first, to convey information concerning entities and relationships which exist in the designer mind; second, to provide information structure, the organization of data where the entities and the relationships are substituted by values.

Information Concerning Entities and Relationships

An entity is an object that can be distinctly identified. Some people may view a certain object as an entity, while other persons may view it as a relationship. Entities are classified into different "entity sets". There are predicates associated with each entity set that

identifies an instantiation of an entity as belonging to that set. If we know that an entity is in a certain set, then we know that it has the properties common to the other entities that pertain to the same set.

A relationship is an association between entities. The "role" of an entity in a relationship is the function that it performs in the relationship. A "relationship set" is a mathematical relation among a number of entities, each of them taken from an entity set.

Table IV shows the most important entities and relationships that exist on the proposed system.

Information Structure

The information about an entity or a relationship may be recorded as pairs of <attribute,data value>, where attribute is a predicate name. If an entity has 5 attributes, for instance, thus we can describe an entity instantiation by 5 pairs of attribute-data value. Note that relationships also have attributes.

Primary Keys. The values of an attribute can be used to uniquely identify an entity in an entity set. The attribute used for this task is

Table IV. Major Entities and Relationships.

Entities	Relationships
Document	Document/Reference
Reference	Document/Annex
Annex	Document/Microfilm
Microfilm	Join (Document-Document)
Thes. Term	Document Log(Doc/Datetime/Depart.)
Department	Holder (Doc/HolderDepart./HolderName)
Holder	Subject Index (Document/Term)
Relationship	Rel. Between Terms (Term/Relat./Term)

known as "primary key". If there is no attribute with unique values we can use two or more attributes. If even using all attributes available in the entity set it is still impossible to uniquely identify each entity, or if simplicity in identifying entities is desired, we can define a new attribute so that such identification becomes possible.

Document Entity Set. Figure 14 shows the DOCUMENT entity set. Note that Document Reference Number (DRN) is an attribute created to simplify document identification. Protocol Number (PROT) could have been elected the primary key. It was not chosen for convenience. The protocol number is large, difficult to memorize, and does not denote the physical place where the document is kept. On the other hand a DRN indicates the document registering year, the document origin (received or sent code), and the chronological sequence number (of receiving or sending). The DRN maps the exact archive where the document is filed. The DRN "89 1 345", for instance, denotes the 345th document received in 1989.

When an entity cannot be uniquely identified by the values of its own attributes we must use ε relationship to identify it. This is known as a "weak entity relationship". The concept of weak entity is also related to the concept of "existence dependency" (Korth and Silberschatz, 1986:29). In Figure 14 the entity set Reference is depicted as a weak entity. If we think that its single attribute, REFER, is a document reference number (DRN) we may conclude that it uniquely identifies a reference. Therefore, it would not be a weak entity. REFER, would be the entity set primary key. The problem is that the reference by itself does not make sense. We are interested in the relationship between a

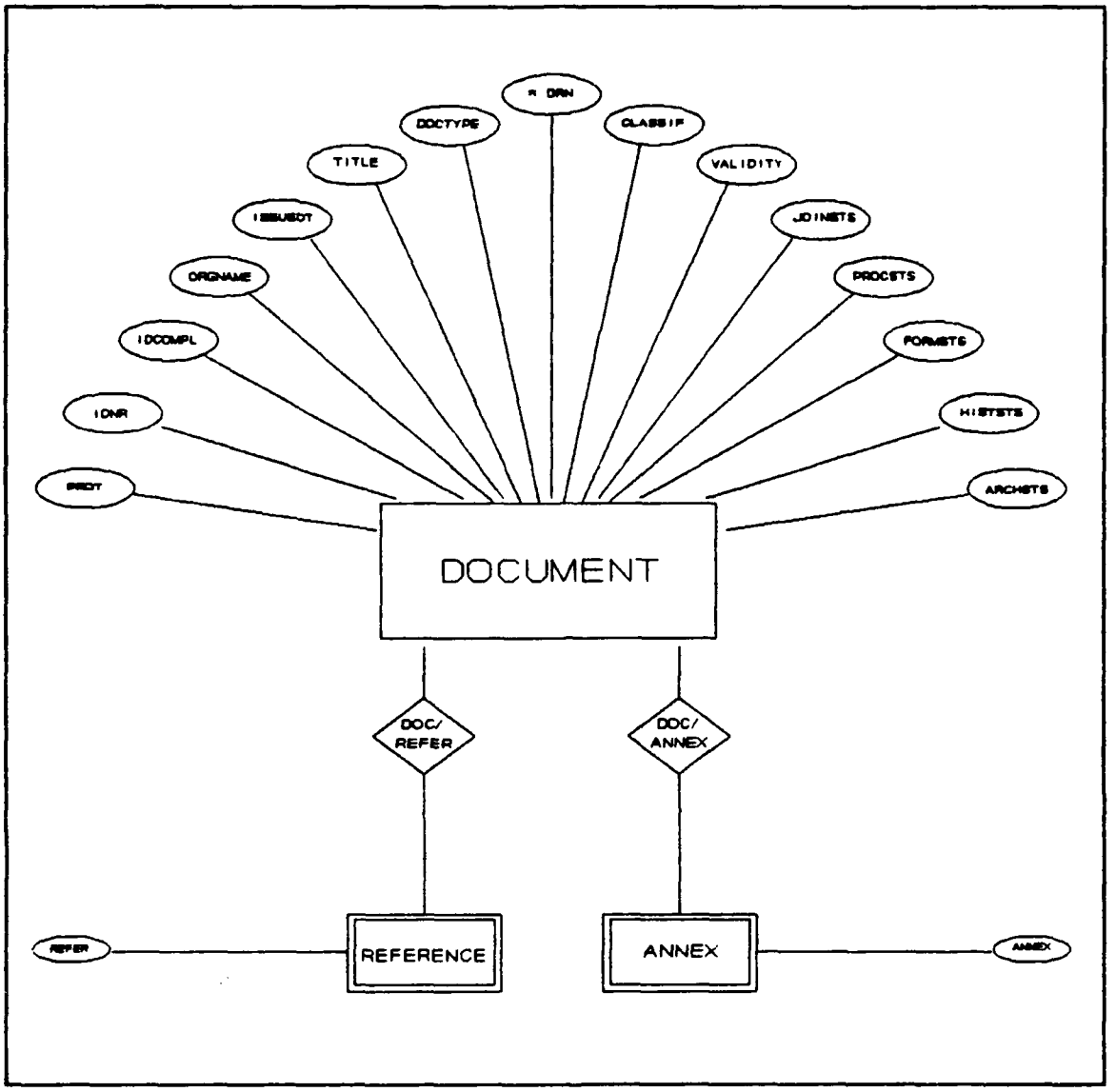


Figure 14. Document Entity Set.

document and a reference. There is no utility in knowing the reference without knowing the document that references it. Furthermore, REFER is a free description of a reference which may consist of a Document Reference Number, the name of a book, or anything else; therefore, it might have homograph references with distinct meanings. For all these reasons the existence of the Reference entity depends on the Document entity.

Moreover, there is no necessity for implementing both the relation corresponding to the Reference entity set and the relation corresponding to the DOC/REFER relationship set. We shall see later (see page 54) that relationship sets derived from the association of a weak entity with a strong entity contain the same attributes as those of the weak entity. For similar reasons, the entity set Annex is also presented as a weak entity.

Join Relationship. The Join relationship is illustrated in Figure 15. The relationship set components are both Document Reference Numbers. Their names, MAIN and JOINED, reflect the "role" they perform. The arrow on the MAIN side means there is a 1:n (n=0, 1, 2, ...) mapping from MAINDOC to JOINED, that is, each main document may have none, one,

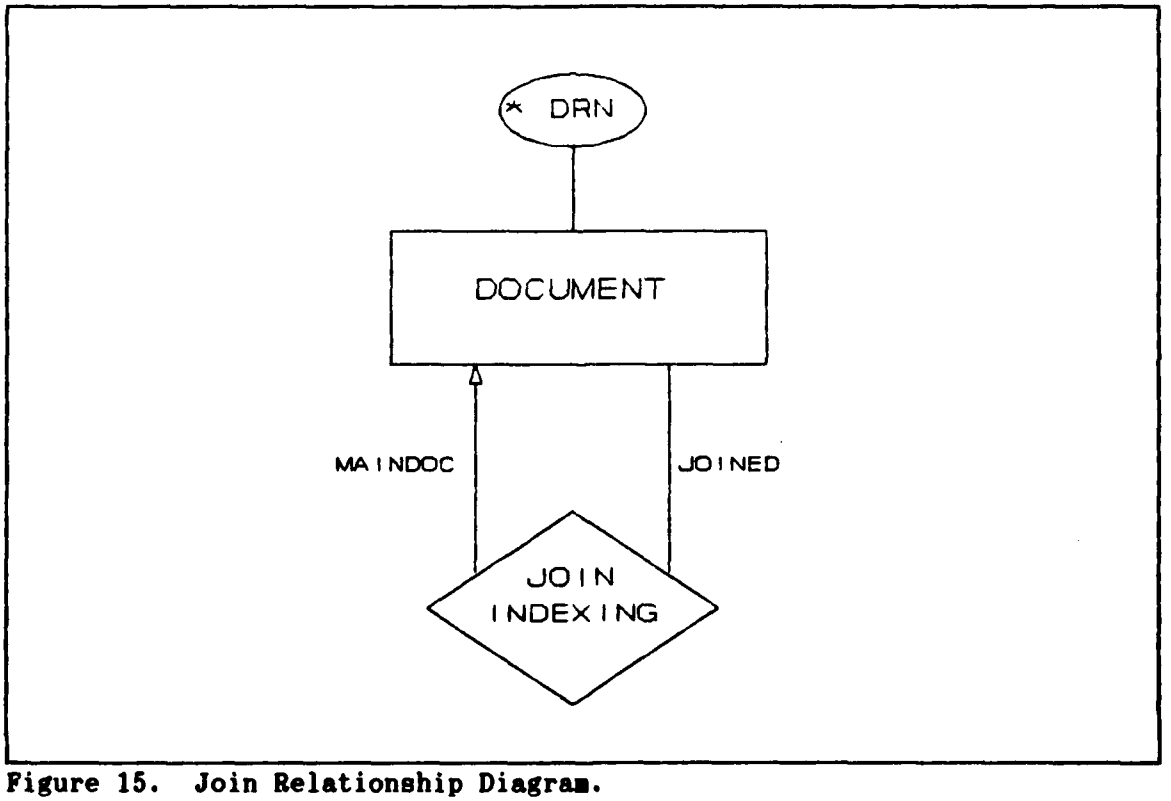


Figure 15. Join Relationship Diagram.

or more joined documents. When no arrow exists on both sides of a relationship the mapping is n:m (n=0, 1, 2, ..., and m=0, 1 , 2, ...).

Microfilm Relationship. Figure 16 illustrates the Microfilm relationship set and the Microfilm entity set. The diagram indicates that each document may have none, one, or more microfilm entities. Each microfilm entity is a frame sequence expressed by the type of microfilm, its number, and initial and final frame number. Although the microfilm entity set possesses a primary key formed by taking the attributes F_TYPE, F_NO, and FIRST (or FINAL) together, it is expressed as a weak entity to indicate existence dependency on the Document entity set. When a document is eliminated from the Document entity set, we also eliminate the relationships on the Microfilm Indexing relationship set that correspond to the document being excluded. Although we do not physically destroy the microfilm, we lose the pointers to the frames.

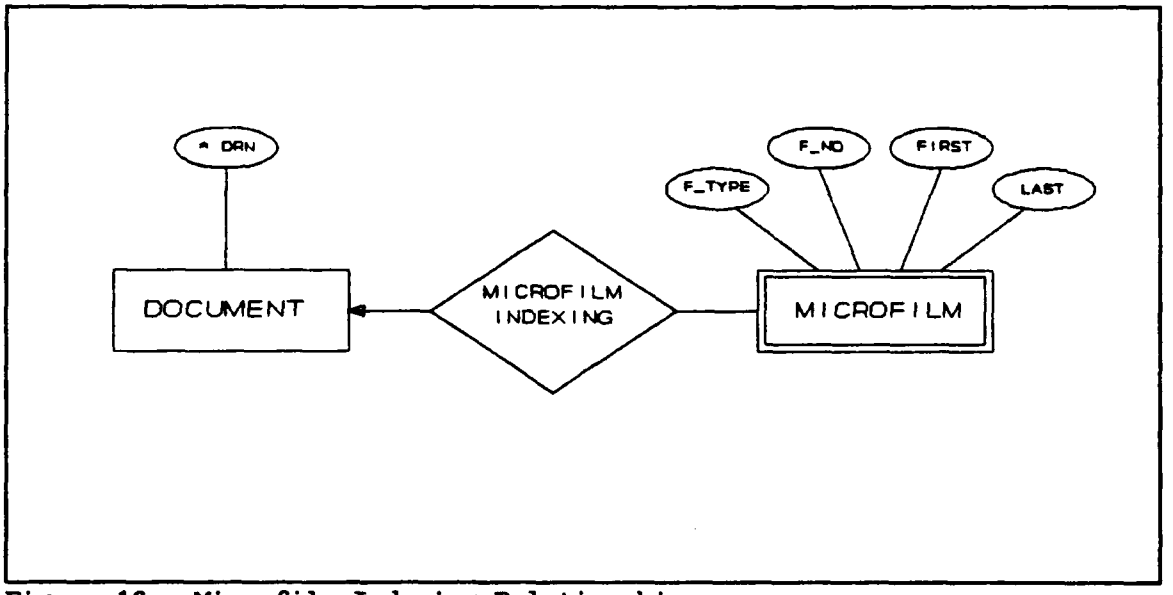


Figure 16. Microfilm Indexing Relationship.

Document and Department Relationships. Figure 17 illustrates the association of the Department entity set to the Document entity set. These entity sets define two relationship sets: Document Log and Holder. On the Document Log relationship set the Datetime attribute guarantees the uniqueness of each relationship and provides means to serialize the

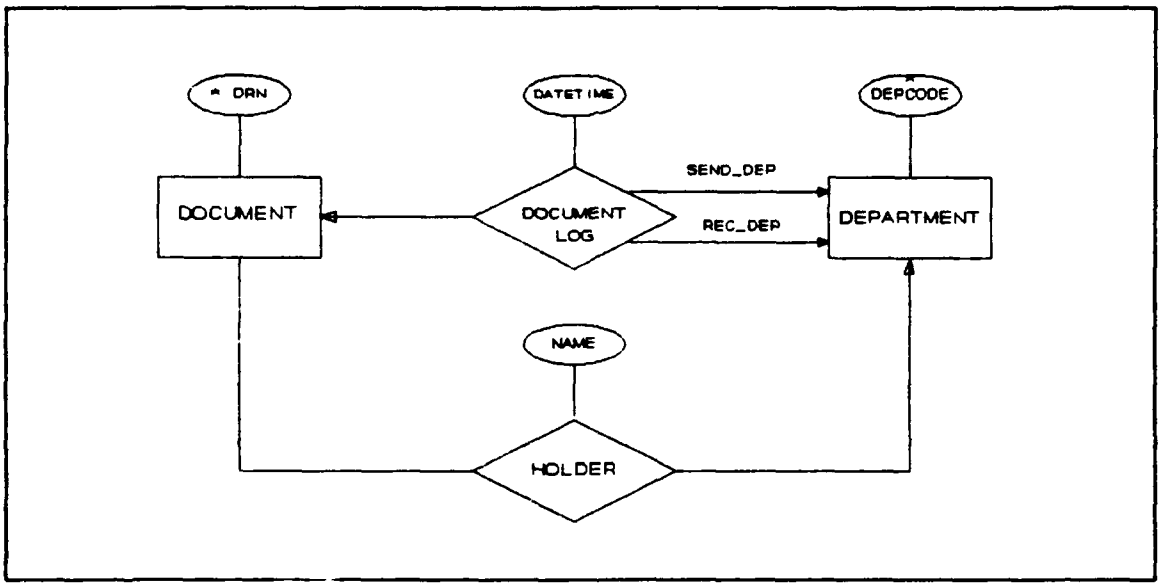


Figure 17. Document History and Department.

document processing. For a given document at each datetime there is only one sending department and only one receiving department. In fact, we shall see later that the primary key of the Document Log relationship is composed only of the DRN, the sender department and the transference date. The situation where a same department sends the same document more than once a day is so rare that it is considered an error. The time attribute has an important role in ordering transferences that were made in the same day. Some constraints cannot be described symbolically. Document transference is made by means of a receipt. It is possible that

transferences are entered in the system out of order. In this situation we have a temporary anomaly. The document log may indicate that someone has sent a document which was not previously received. We want to be able to recognize this inconsistency and to recover when the missing record is entered.

The second relationship set, Holder, permits the identification of a department code and a holder person name which are responsible for a document. The diagram shows that the name of the holder is not directly related to the department set. It was designed to permit a department to keep its own control over the personnel actually processing documents without bringing unnecessary complexity to the system (see Document History, page 20). Therefore, there is neither the necessity of entering this information on the system (it is optional) nor prior registering the name that is going to be entered.

Document-Thesaurus Relationships. Figure 18 shows the entities and relationships related to the thesaurus The single attribute of a thesaurus term is its class. The primary key of the term entity set is the term name. The class logically divides the terms in related subjects. A class entity set, not shown in the illustration, is used for data validation purposes. It was decided to use the term name as the primary key, instead of the term name and the class. As a consequence, the term name must be unique. It was chosen for the sake of simplicity. The occurrence of duplicated terms does not justify having to specify a term class in all term relationships and queries. As we do not expect a large number of duplicates, the simplest solution is to add the class

code to the name itself, to differentiate from an existing homographic term.

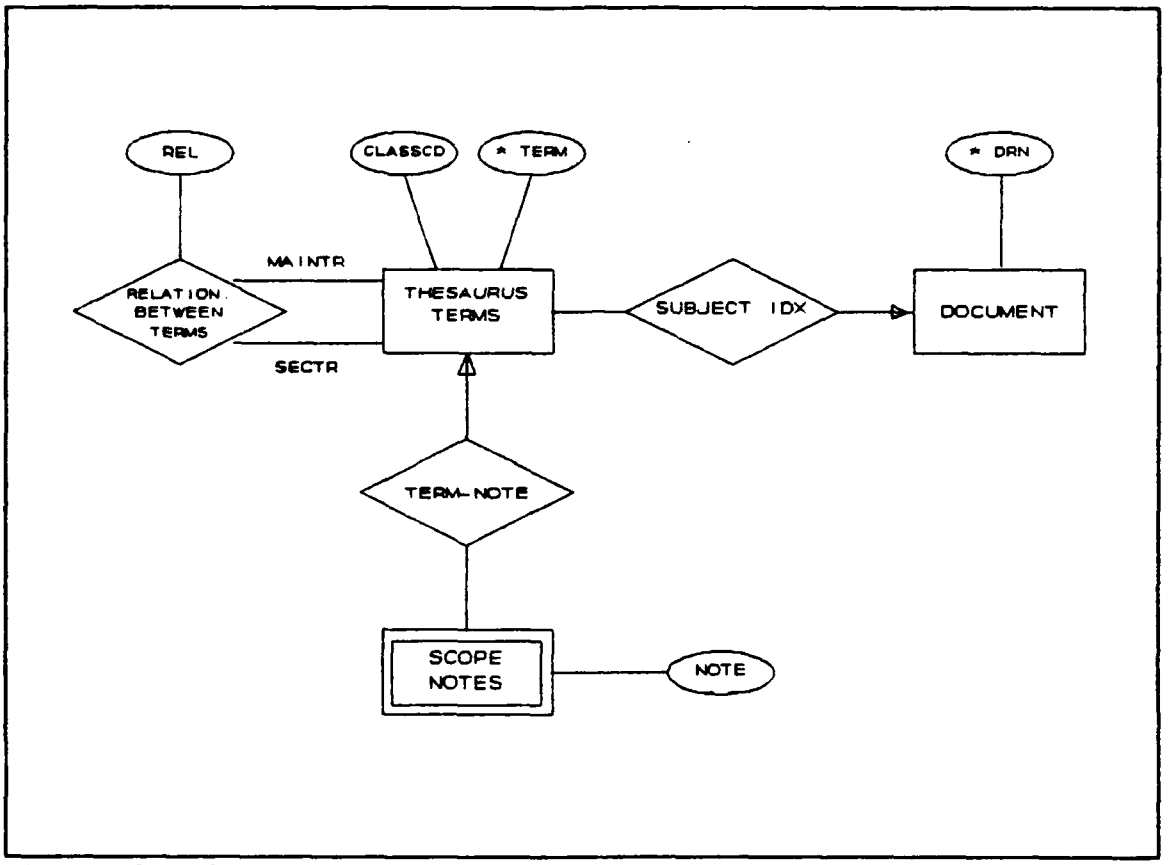


Figure 18. Thesaurus-Document Entities and Relationships.

The first step in defining a new term is entering the term description on the Terms entity set. We may supply the user with usage details by creating a scope note associated to the entered term. The next step is defining relationships between the new term and other existing terms. Different consistency constraints apply for each relationship between terms (REL). These constraints are shown in Appendix C, page 121.

Reducing E-R Diagrams to Tables. The E-R diagrams can be represented by a collection of tables. Table V shows the tables derived

Table V. Tables Derived from the E-R Diagrams.

DOCUMENT (DRN, P_SC, P_OC, P_NUMBER, P_YEAR, IDNR, IDCOMPL, ORGNAME, ISSUEDT, TITLE, DOCTYPE, CLASSIF, VALIDITY, JOINSTS, PROCSTS, FORMSTS, HISTSTS, ARCHSTS)

REFERENCE (DRN, REFER)

ANNEX (DRN, ANNEX)

JOIN (MAIN, JOINED)

MICROFILM (DRN, F_TYPE, F_NO, FIRST, LAST)

DEPARTMENT (DEPCODE)

HOLDER (DRN, DEPCODE, NAME)

DOC_LOG (DRN, DATE, TIME, SEND_DEP, REC_DEP)

TERMS (TERM, CLASSCD)

CLASSES (CLASSCD, CLASS)

RELSHIPS (MAINTR, REL, SECCOD)

NOTES (TERM, NOTE)

SUBJECT (DRN, TERM)

from the previous diagrams. As a general rule, for each entity set and for each relationship set in the model we define an unique table. In tables derived from entity sets, the columns correspond to attributes in the former entity set. In tables derived from relationship sets, the columns correspond to those attributes that constitute the primary keys of the originating entity sets. Weak entities are an exception to the general rule. The table derived from a weak entity contains all attributes of that entity plus the attribute(s) that constitutes the primary key of the strong entity on which the weak entity depends. The

Reference, Annex, and Microfilm entities are examples of weak entities. We observe on Table V that those tables have the document reference number (DRN) as an attribute. In this case, the tables derived from the relationship set and from the entity set are equal. Therefore, we implement only one of them. The columns that constitute the primary key of each table are underlined. A table that has no underlined columns has the primary key composed of all columns.

Figure 19 illustrates the tables needed to compose the information about a document. Note that a joined document has only one main document while a main document may have several joined documents. The document in the figure has been microfilmed. Therefore, it must have the "form status" set to "M" of microfilm. If it were a destroyed document it would have only the document and reference entries. The reference would be pointing to the document that authorized its destruction, while the form status would contain the value "D" of destroyed document. Since the data model is complete, next chapter presents its implementation.

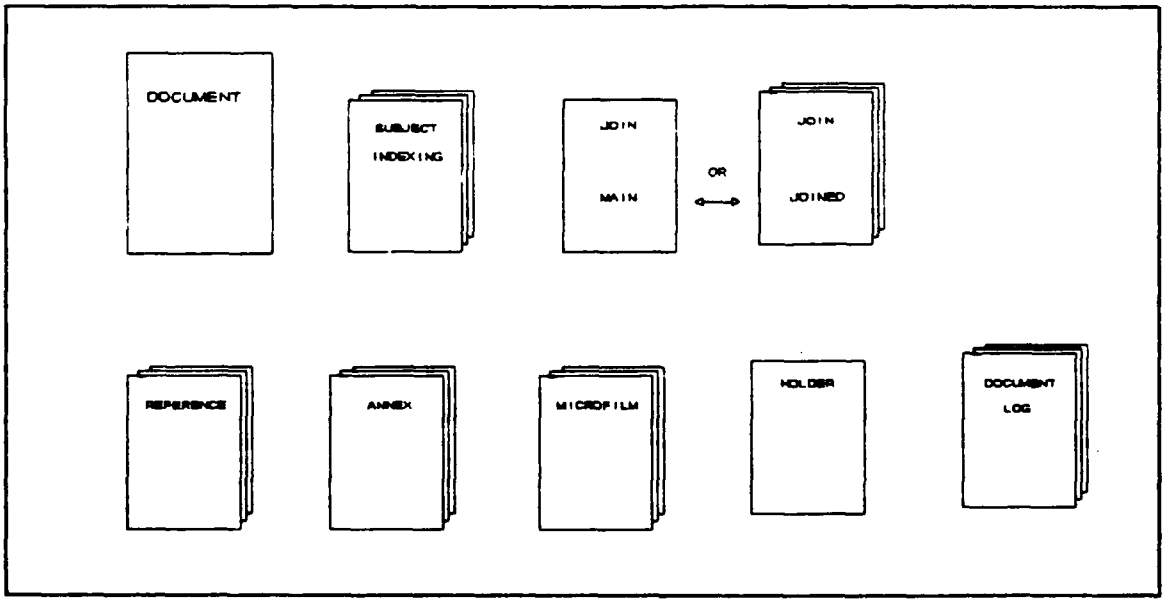


Figure 19. Tables that Compose a Document.

V. Implementation

Introduction

This chapter addresses the implementation of a prototype of the proposed system. The term prototype is used in a sense of "a first system" that still has to be evaluated by the user and refined to reach its final form. It does not mean a "working prototype" from the "prototyping" software paradigm, which is used only as a tool to refine requirement specifications. A "working prototype" may even use a inappropriate operating system or programming languages because it is mainly concerned with a quick development for demonstrating capabilities. The prototype developed in this thesis effort is not an adaptation of program fragments intended to show system potential but a full implementation of all specifications presented in the previous chapters. We present the steps, how they were accomplished, the reasons, facilities and difficulties. First we talk about fourth-generation languages in general, and about SQL language and Oracle's Relation Data Base Manager System (RDBMS) in particular. We divided the implementation into several steps, each discussed in detail. Additional information about system files, field specifications, screen templates, SQL statements, and triggers are given in the appendices.

Forth-Generation Languages - 4GL

Fourth-generation languages are consequence of the need to instruct computers more easily and more quickly than in the past. First generation languages' characteristic was the binary notation (ones and

zeros), since early computers did not have interpreters or compilers to translate machine language to a more readable form. The second generation languages used symbolic assembly languages where the physical address of variables and their names became independent. Third generation languages have been used since 1960 and are referred to as "high-level" languages. They have been continually improved and still are extensively used. Some of their most important aspects are the use of English words, mathematical notation, and the hardware independence, which is provided to a certain extent. The major drawback of third generation languages is the time consuming need for writing and debugging a number of lines of code, which makes the modification of complex systems a very difficult task.

Forth-generation languages vary greatly in their power and capabilities. The term encompasses software tools that enable the software developer to specify some particular aspect of the software at a high level. The tool then automatically generates the code, based on the user specification. Therefore, while some 4GLs are merely query languages or report generators, others are able to build a complete application. Some 4GLs are referred as "nonprocedural languages". A nonprocedural language specifies "what" is needed, instead of focusing on procedural details of "how" to accomplish the action. SQL - Structured Query Language is a relational database language that is an example of nonprocedural language. The following statement is a complete program in this language:

```
SELECT DRN, TITLE
FROM DOCUMENT
WHERE ORGNAME = 'AFIT'
```

```
AND ISSUEDT BETWEEN
'10-JAN-89' AND  '20-FEB-89';
```

Oracle RDBMS uses SQL as its data manipulation language and offers additional facilities that are described in the next section.

Oracle's Development Tools

Pressman states (Pressman, 1987:24) that 4GL environments offer some or all of the following tools: "nonprocedural languages for data base query, report generation, data manipulation, screen interaction and definition, and code generation ..." Thus, Oracle does offer a 4GL environment. According Lans, Oracle's SQL (SQL*Plus) has a "noteworthy" large number of additions (Lans, 1988:283). Besides the facilities offered by the language itself Oracle provides SQL*Forms, a tool designed to simplify system/user interaction, and SQL*Reporter, a tool to generate printed documents derived from the database. The main concepts of SQL*Forms are shown in Table VI. Form, page and block concepts are explained in the next paragraphs. SQL statements were already introduced but they will be illustrated again, later, together with triggers and macros. The Reporter facility will be commented on the paragraphs about the modules Print Processing Sheet and Print Thesaurus Terms.

Form. A form is a screen layout that presents a fill-in-the-blanks arrangement of database information. It permits efficient data entry, update and query that, otherwise, would require the operator to use SQL statements. The Oracle program that creates and maintains forms is referred as SQL*Forms. We may see a form as a independent set of specifications that works like a program. These specifications include screen definitions, tables, queries, integrity rules, data validation,

Table VI. SQL*Forms Main Concepts.

- form
- block
- page
- SQL statements
- triggers
- macros

etc, which are made in a interactive manner by typing statements in "popup" windows. Forms can call each other, but only can exchange information by explicitly defined global variables. A form has one or more blocks and one or more pages. The smallest form would have at least one "page" and one "block." A form is saved inside the database files and also externally in a ASCII file that has a .INP extension. Applications forms can migrate to a different computer system that also uses Oracle RDBMS using these files.

Block. A block is a subset of a form that provides automatic insertion, deletion, update, and query to a table on the database system. We can think of a block as a standard procedure available in a program which can be used as many times as needed, and tailored to fit our necessities. A block can also access no tables. An example is a form used as a menu. The single block only contains a control variable that triggers macro statements calling the "subordinate" modules. Note that "subordinate module" only means a module that is called by another module, which does not imply any other kind of interaction.

A block has a "base table" but can contain also fields from other tables. Automatic operations are supported only for the base table.

Those fields are manipulated by specifying triggers and SQL statements to act on them. These "extraneous" fields are commonly used to access and/or display related information, and for complex data validation and integrity enforcement. Information can be exchanged among blocks by referring to a field in a "select into ..." statement, or by using a macro statement as "#COPY from_field to_field".

Page. A page is the part of the form which the user sees on the screen. A form may have as many pages as are necessary. A block may occupy only part of a page, an entire page, or several pages, according to the necessity and the extension and number of fields on the base table. We may have also more than one block on a single page. Since each block has its own base table, they act independently of each other. It is possible to create for the user an illusion of interdependency by using triggers and macros, which simulate the operator strokes by executing hidden statements. Later in this chapter, some examples are given.

Implementation Strategy

It was necessary to define an implementation strategy to guide the development effort. The steps of this strategy are described in Table VII. Each one of them is explained and discussed in the following paragraphs.

Creation of Database Tables, Clusters and Indexes

The implementation begins by creating the tables that will constitute the database. Additional tables were created for data validation and integrity rules.

Table VII. Steps of the Implementation Strategy.

1 - Creation of database tables, clusters and indexes.
2 - Development of a experimental module.
3 - Design of screen templates.
4 - Distribution of screens within the forms.
5 - Implementation of basic functions and simple data validations.

We added to the tables shown in Table V the tables SEQNUMBERS, RELNAME, and CLASSIFICATION. The table SEQNUMBERS is used to generate Document Reference Numbers automatically, the table RELNAME contains the relationship codes and complete names, and finally the table CLASSIFICATION that contains the classification codes with their respective complete names. In the preceding chapters we have concentrated on the conceptual level. However, a major issue is the system performance. The performance depends not only on the efficiency of the data structures used to represent the data in the database but also on the system efficiency in accessing and manipulating these data structures. Clustering and Indexing were used to improve the system performance and are explained in the following paragraphs. The file used to create the tables and insert initial data is shown in Appendix A.

Clustering. This is a database technique and Oracle feature that permits both access time and disk storage space savings.

Access Time. Data is transferred between disk storage and main memory in units of storage called "blocks". Clustering saves access time by organizing related information in a single block and, when a single block is not enough, in contiguous blocks. The performance of join queries is improved because rows that are joined are stored

together. Usually, the rows of a table are represented by records in a file. If we assign the records randomly, it can be the case that a different block must be accessed for each row needed.

Disk Storage Space. We may be able to save disk storage space by storing only once values of similar columns that were defined in different tables. Therefore, to be clustered, a group of tables must share a column with the same type, length, and meaning. Such a column is called "cluster column". Clustering affects the way tables are physically stored on disk, while having no effect on the logical appearance of the table. Therefore, it is not necessary to know the name of a cluster column in order to use it in a query. The program will use it automatically, whenever the query that was requested permits its use.

Table VIII shows the tables that were clustered on the Document Reference Number column (DRN) and on the Term column (TERM). All

Table VIII. Clustered Columns and Tables.

Cluster Name	Columns	Tables Included
C_DOCUMENT	DRN	DOCUMENT REFERENCE ANNEX JOIN MICROFILM DOC_LOG HOLDER SUBJECT
C_TERM	TERM	TERMS RELSHIPS NOTES

information related to a single document are kept together providing a minimal access time.

We decided to cluster the Subject table by the column DRN instead of clustering by the column TERM because join queries using DRN will be more frequent. Suppose we are interested in documents about "jet engine" and "maintenance". The query has to find all the documents indexed by "jet engine" and then verify whether those documents are also indexed by "maintenance". If we suppose the query accesses 1000 documents registered under the term "jet engine" and that each document is indexed by 5 terms in average, the total number of accesses by TERM key would be 1000, and the total number of accesses by the DRN key would be 5000. Actually, the number of accesses is smaller than 5000 because the subquery stops when a positive comparison is reached.

Sometimes a cluster that contains just one table also saves disk storage space. This would be the case of the table Terms, where a same value of CLASSCD is repeated on the rows of all terms pertaining to that class. Since nested clustering is not possible, in place of clustering by CLASSCD we preferred to keep together all information related with a same TERM. In addition to the organizational advantages of clustering, the DBMS creates an index on the clustered column(s) that increases the access speed.

Indexing. An index to a table helps to find information quickly. The program can "look up" the rows in an index to the table. Otherwise, it would have to scan all the rows of the table. Indexing a table may reduce the time requested to perform a query, mainly if the table is large. On the other hand, if the table has a few columns (a few may be

considered as less than two hundred) the overhead involved in using an index will probably exceed the time saved. Another use of indexing is to guarantee that a column of a table contains unique values (as we want for the primary key of a table), although we did not use the index feature on this purpose (we used the "primary key" feature of SQL*Forms). By contrast, clustering does not enforces uniqueness. Therefore, we used the indexing technique on the columns expected to be frequently accessed by queries, whenever clustering was not possible, not necessary, or not efficient enough. We also used indexing on the columns where speed is more important than saving disk storage space.

Table IX shows the indexes that were created on the database tables. Alike clustering, the user does not need to know whether a column is indexed, nor the name of a index. If an index exists, the query will use it automatically, whenever possible.

Table IX. Indexed Columns and Tables.

Index Name	Column	Table Name
I_TERM	TERM	SUBJECT
I_P_NUMBER	P_NUMBER	DOCUMENT
I_IDNR	IDNR	DOCUMENT
I_ORGNAME	ORGNAME	DOCUMENT

Development of an Experimental Module

This step was to help me, as a new Oracle developer, to learn the capabilities of the Oracle system and to choose a suitable style to be used on all the other modules. The Enter Document module was chosen to be the experimental module. In this phase I became familiar with defining forms, defining database and control fields, choosing field attributes, establishing data validation, using triggers, and using SQL statements inside a form. SQL*Forms does not provide means for copying trigger steps from one trigger to another one, nor triggers from one block to another block in the same form. This ability is useful when standardization of commands and messages is needed.

To overcome this deficiency, part of the work was done directly on the INP files. An INP file (.INP suffix) is an intermediate text file that contains a form description used by the system to generate form files (.FRM suffix), to load the form description into the SQL*Forms program, or to convert the form description into a database format (which is kept in several Oracle system tables).

After doing the modifications, the program Interactive Application Converter - IAC was used to convert the INP file into a database format. The program Interactive Application Generator - IAG was used to generate the form file. Another possibility investigated was a memory resident text editor to cut and paste on the SQL*Forms Paint Screen. This process permits SQL statements to be copied between fields and blocks and also between forms. Its disadvantage is the abrupt end of the SQL*Form program, losing the modifications already done, when it does not have enough memory space to keep those changes. To avoid this loss it is

necessary to save the form, exit from SQL*Forms, and enter again. Therefore, due to the memory space used by the resident program, a user should save the work frequently.

Design of Screen Templates

Table X shows the conventions used in template design. The screens were designed outside SQL*Forms, before creating the forms that would use them. They were designed using a text processor.

Figure 20 shows an example of a menu screen used in the system. To give the user the notion of which level he or she is working, it used a windowing like menu system. Some menus go one level deeper by having another submenu window that partially covers the previous menu.

The system has around 30 different screens (or pages) and the style used for all of them has the following characteristics:

- use of a windowing menu tree
- a single-line box surrounding no menu screens
- enumeration of blocks in a same screen
- separation of blocks in a same screen by a double line
- module name on the left bottom corner of the screen
- main keys used on the module at the bottom of the screen

All screens are shown in Appendix B.

Distribution of Screens Within the Forms

To facilitate the identification of forms, blocks, and pages, the method that follows was used in naming the screen templates. Each screen represents a module of the proposed system (see Figure 11). The screen name is the code that corresponds to the system module (ex: A0, A34,

Table X. Symbols Used in the Screen Templates.

C	fields may contain any combination of letters, digits, blank spaces, punctuation, and special characters.
9	field may contain any number.
DD	day of the month.
MON	month of the year, as JAN for instance.
YY	last two digits of the current year.
HH	hours of the day, in a 24 hours format.
MI	minutes of the hour.
()	enclosing a field definition, as "(CCCC)" for instance, to represent fields that cannot be accessed by the operator. Protected field.
->	at one end of a field, as "CCCC->" for instance, or above the field name, to indicate that the field length is longer than the display length. The operator can use the arrow key to scroll the field horizontally.

etc). Each screen has its name written on the left bottom corner of the template. Therefore, each screen template also corresponds to a "page" of a "form" on the Oracle system. When a form has only one page (it corresponds to just one system module) the name of the form is identical to the name of the screen it represents. When the same form has more than one page (more than one screen) two situations may arise:

1. Each page constitutes a different module. The form borrows the name of the first page. Usually, the first page corresponds to a "parent" module and the other pages to "children" modules, with the same structure of the proposed system.

2. A same module spans more than one page (never more than two pages). Each screen has the same basic name but differs on suffix. The first page has a suffix "-1" and the second page has a suffix "-2".

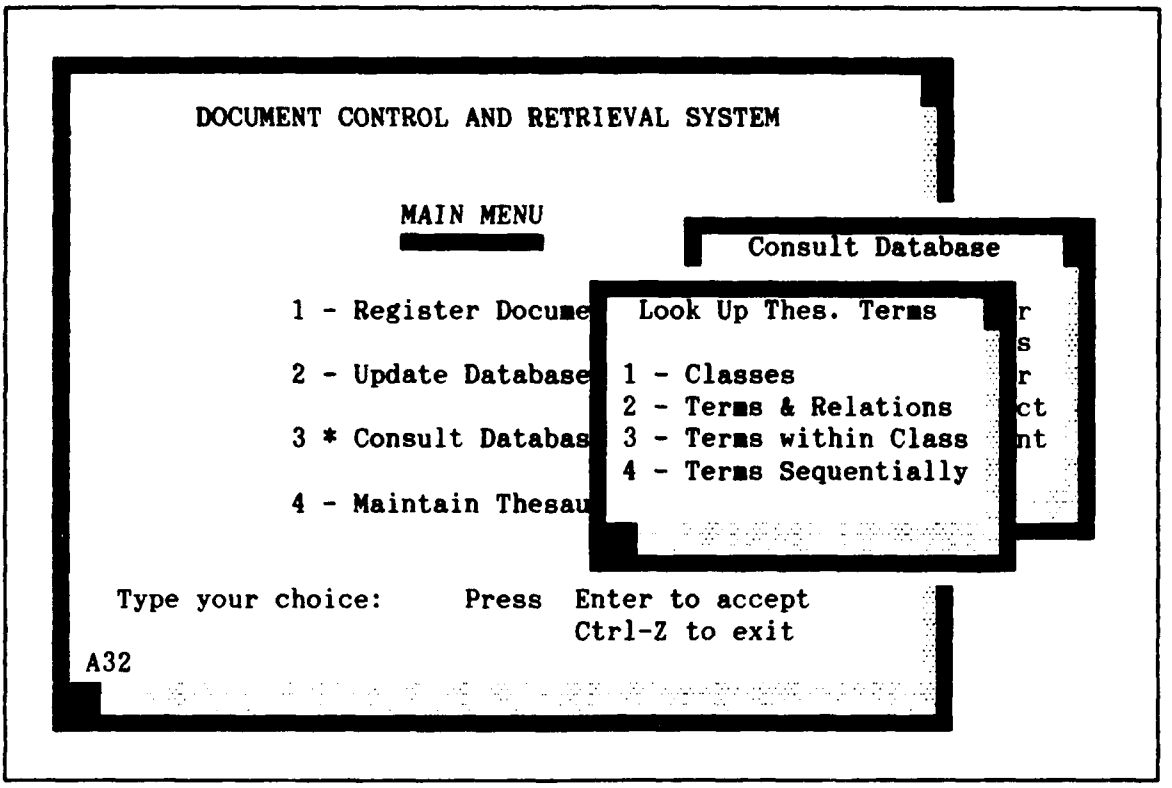


Figure 20. Example of Three Level Menu.

A page may contain one or more than one blocks. Blocks that have a base table are referred to by the base table name. Blocks that do not have a base table are usually alone in a page and are referred to by the module name which they represent.

The criteria used to place modules (pages) in the same form was:

- necessity of communication between modules. A block placed in screen "A" may need to copy the value of a field which is placed somewhere in screen "B". Blocks communicate more easily when placed in the same form. Communications between forms have to be done using global variables that have a constrained use.

- necessity of reducing the loading time of a called module. Some modules are expected to switch frequently from one module to the other. We shall reduce the waiting time as much as possible. This is the case of the menu tree which was placed in the same form to reduce the time between displaying the main menu and a submenu.

The criteria used to place modules in distinct forms was:

- independence, that is, there is no need of accessing a field pertaining to another block in a different screen. The module is a complete and functional unit.

- simplicity. Although a module may logically correlate to other modules, if there is no physical dependency, making it a single-page form simplifies maintenance.

The arrangement summarized in Table XI was obtained by applying the heuristics described above. The table maps system modules to pages of a form. Some forms, as form A0 for instance, encompass more than one module. In form A0 each module is a page. On the other hand, a page may encompass more than one block, as form A241 page #1 for instance. Coordinate or control blocks, which do not have a base table, are indicated in the table by a star preceding the block name. Form A0, in spite of having several screens and blocks, does not have any "base table". The same happens with module A36. In spite of accessing several tables to calculate the statistics, it does not have a base table.

Implementation of Basic Functions and Simple Data Validations

For the sake of simplicity, the implementation of forms was divided into two phases. In the first stage the forms were created but without the use of triggers and imbedded SQL statements, except when they were strictly necessary to implement basic functions. Table XII lists the tasks done in this phase.

Implementation of Complex Data Validations and Integrity Rules

Integrity constraints and validation criteria were condensed by module to facilitate the forms development and to provide means for their testing. Constraints were listed according to significant input conditions. Some of these conditions were Insertion, Update, Deletion

Table XI. Screen Distribution within the Forms.

Module Name	Form	Screen	Pg#	C	Block(s)
Main Menu	A0	A0	1	*	A0
Register Document	"	A1	2	*	A1
Update Database	"	A2	3	*	A2
Transfer Custody	"	A25	4	*	A25
Consult Database	"	A3	5	*	A3
Look up Thesaurus Terms	"	A32	6	*	A32
Maintain Thesaurus	"	A4	7	*	A4
Enter Document	A11	A11	1		DOCUMENT, REFERENCE, ANNEX.
Print Processing Sheet	A12	A12	1	*	A12
Generic Update	A21	A21-1	1		DOCUMENT REFERENCE, ANNEX
		A21-2	2		HOLDER, DOC_LOG
Change Validity	A22	A22	1		DOCUMENT
Process Document Destruction	A23	A23	1		DOCUMENT
Join Documents	A24	A24	1		JOIN
Transfer Department Custody	A251	A251	1		DOC_LOG
Transfer Holder Name	A252	A252	1		HOLDER
Update Microfilm Data	A26	A26	1		MICROFILM
Subject Indexing	A27	A27	1		SUBJECT
Browse by Holder	A31	A31	1		HOLDER,DOCUMENT
Look Up Classes	A321	A321	1		CLASSES
Look Up Terms/Relationships	A322	A322	1		TERMS,RELSHIPS
Look Up Terms within Classes	A323	A323	1		TERMS
Look Up Terms Sequentially	A324	A324	1		TERMS
Browse by Header	A33	A33	1		DOCUMENT
Browse by Subject	A34	A34	1		SUBJECT,DOCUMENT
Retrieve Document	A35	A35-1	1		DOCUMENT,SUBJECT
		A35-2	2		REFERENCE,ANNEX MICROFILM,JOIN, HOLDER,DOC_LOG.
Show Statistics	A36	A36	1	*	A36
Maintain Classes	A41	A41	1		CLASSES
Maintain Terms	A42	A42	1		TERMS,NOTES
Maintain Relationships	A43	A43	1		RELSHIPS
Print Thesaurus Terms	A44	A44	1	*	A44

and Query. Appendix C shows the integrity rules, Appendix B the screen templates where the rules apply, and Appendix D some of the triggers that implement the integrity rules. Actually, Appendix D does not encompass all existing triggers, due to their great number. Those listed are

Table XII. Tasks performed in the Implementation of the Basic Functions.

- Creation of forms, blocks and fields;
- Specification of field attributes using the Specify Attributes feature of SQL*Forms;
- Data validation of field input, using the Specify Validation feature;
- Definition of the default ordering in queries using the Default Ordering Window;
- Specification of triggers, using the Trigger Step feature.

samples of a class of integrity enforcement.

To illustrate the accomplishment of integrity rules, an example was selected from module A251 - Transfer Custody - Department. The following integrity rule is listed under the insertion input condition (Appendix C, page 114):

- Set the document history status equal "A" (alteration) if there exists a transference of the document in which the sending department is different from the receiving department in the previous row. Otherwise, set the document history status equal "N" (normal). Example: assume there are two rows in the Doc_Log table, ordered by DRN, Date, and Time, in ascending order, with these values:

DRN	Date	Time	Send_Dep	Rec_Dep
99999999	DATE1	TIME1	DEP1	DEP2
SAME DRN	DATE2	TIME2	DEP3	DEP4

The underlined departments should be the same. As they are not the same, set the document history status equal "A".

Figure 21 shows the trigger step that verifies the rows corresponding to the document being transferred, in the table DOC_LOG, to enforce the integrity rule described above. The trigger step was extracted from the A251.INP file that describes the form A251, and is

```
*POST-INSERT
;SQL>
SELECT *
FROM DOC_LOG Y
WHERE EXISTS    (SELECT *
                 FROM DOC_LOG X
                 WHERE DRN - :DRN
                 AND Y.DRN = DRN
                 AND Y.SEND_DEP <> REC_DEP
                 AND
                     TO_NUMBER(TO_CHAR (Y.LDATE,'J')
                     || SUBSTR(Y.LTIME,1,2)
                     || SUBSTR(Y.LTIME,4,2))
                              =
                     (SELECT MIN (ALL  TO_NUMBER(TO_CHAR (LDATE,'J')
                                       || SUBSTR(LTIME,1,2)
                                       || SUBSTR(LTIME,4,2)))
                      FROM DOC_LOG
                      WHERE DRN = :DRN
                      AND TO_CHAR(LDATE,'J')
                          || SUBSTR(LTIME,1,2)
                          || SUBSTR(LTIME,4,2)
                               >
                          TO_CHAR (X.LDATE,'J')
                          || SUBSTR(X.LTIME,1,2)
                          || SUBSTR(X.LTIME,4,2)
                     )
                )
/
;Message if value not found :
$SET_ALTERATION $SET_NORMAL
```

Figure 21. Trigger Step to Validate a Document Transference.

also listed in Appendix D, page 146. The step is part of a "Post-Insert" trigger. This means that it is executed after the operator has inserted the new row in the table Doc_Log, corresponding to the transference being made. Observe the last line of the step. The first "$" symbol indicates the label of the next step to be performed if this step succeeds. The second "$" indicates the failure label. The step shown does not set the document history status itself; but, if there exists a row where the

sending department is different from the receiving department of the previous row the step succeeds and the success label deviates the trigger to the step that will set the status equal "A"lteration. On the other hand, even if the history status was alteration, if the insertion of the present row brings consistency to the document log (suppose the missing row was introduced) the step fails, causing the trigger to set the history status "N"ormal. Therefore, this step accommodates transference insertions that are made out of order.

The SQL query that analyses the rows is one of the most complex queries in the entire system. Note that it gives to the table Doc_Log the "aliases" Y and X to make comparisons with distinct instantiations of the table Doc_Log. The logical expression "exists" is evaluated as true if the subquery returns at least one row, and false if not. The date (LDATE) is transformed to Julian date and is concatenated to the time (LTIME) without the comma. This makes comparisons between date-times easier.

The analysis is made by selecting each row of the document being transferred. This row is compared with the row that has the "MIN"imum date-time value among those rows that have the date-time value greater than the row that was selected for the comparison. The comparison verifies that the Send_Dep of the selected row is different from the Rec_Dep of the "greater's-smallest" row. This way, each row is compared with the row that has the date-time value immediately greater.

Implementation of Printed Reports

Although all queries displayed on the screen may be output to a printer attached to the microcomputer terminal there are two forms specially designed to generate printed reports. They are the A12, Print Processing Sheet and the A44, Print Thesaurus Terms. While it is easy to extract reports using line commands in SQL*Plus, we need standardized reports in special situations. The first is the processing sheet that is attached to each document, the second is the printed thesaurus dictionary, which may be used for off-line document indexing. These reports are generated using the Oracle tool SQL*Report, a procedural system composed of two programs. The "Report Generator - RPT", is used to extract information from the database and the "Reporter Text Formatter is used to format the report. To generate a report we create a source file which is successively compiled by both programs. Appendix E lists the object file of the processing sheet and two of the four reports generated for the thesaurus. The remaining object files are not presented because they are only variations of the other ones. Note that these reports are executed from inside SQL*Forms with no need of quitting the program. The reports are printed simultaneously to the printer and to disk files. The disk files have the same name of the generating Form concatenated with the corresponding menu number, and a suffix LIS. Therefore, the program A11 which gives report options 1, 2 and 3 prints reports to the files A11_1.LIS, A11_2.LIS and A11_3.LIS. Each time the same report is generated it overwrites the previous report. Figure 22 gives a sample the format of the report A44_3.LIS which orders terms by their hierarchical relationship. For demonstration

```
 THESAURUS REPORT A44_3 - ORDER BY TERM HIERARCHICAL ASSOCIATION
08/03/89 03:53                                                PAGE 1

   Level    Term

   1. ROOT
      2. DOCUMENT CONTROL
         3. A0
            4. A1
               5. A11
               5. A12
            4. A2
               5. A21
               5. A22
               5. A23
               5. A24
               5. A25
               5. A25
                  6. A251
                  6. A252
               5. A26
               5. A27
            4. A3
               5. A31
               5. A32
                  6. A321
                  6. A322
                  6. A323
                  6. A324
               5. A33
               5. A34
               5. A35
               5. A36
            4. A4
               5. A41
               5. A42
               5. A43
               5. A44

                                END OF REPORT
```

Figure 22. Report A44_3.LIS, Terms by Hierarchical Association.

purposes, the names of the modules that compose the proposed system were entered as thesaurus terms, creating the same hierarchical relationships that exist in Figure 11. The report traverses the hierarchical tree in

preorder; giving the term name and the term level in the hierarchical relationship tree.

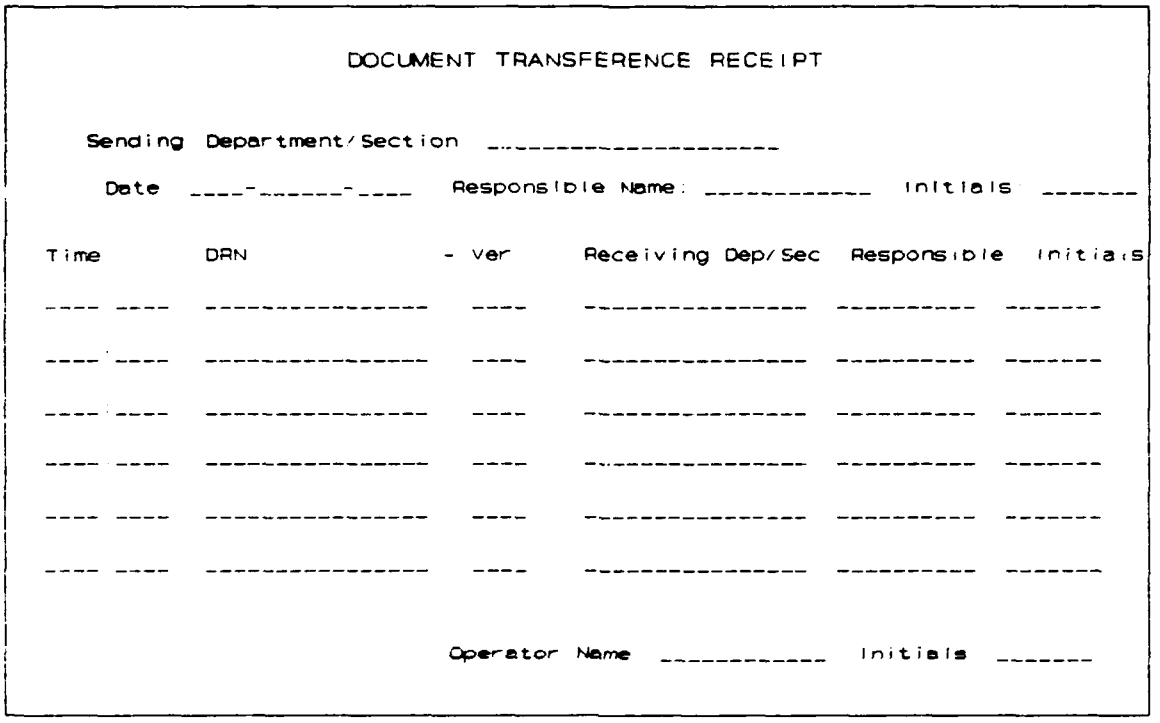
Design of the Document Transference Receipt

When a document is transferred from one department to another the updates of tables Holder and Doc_Log are made using the module A251, Transfer Custody -Department. This task may be assigned to the Protocol Section, to the Archive Section, or even to both of them. Whatever section is chosen, they cannot afford to have a computer station dedicated only to enter document transferences. But, because we do not want to introduce delays in the document processing, we create a document transference receipt.

The proposed model is shown in Figure 23. The sending department may place in one receipt more than one document. Also, documents in the same receipt may have different destinations. Someone in the transferring department fills the receipt fields except those corresponding to the transference time, receiver name, and initials. The distribution man distributes the documents, collecting the receivers' names and initials. He also enters the Time the transference was made. Upon completion, he leaves a copy in the section assigned to enter document transferences and keeps the original with him, after noting the name and initials of the transference station responsible.

Integration and Testing

Two kinds of tests were performed: individual test and integrated test. The first one was concerned with the proper execution of the fundamental operations. The second one, executed after integrating the



DOCUMENT TRANSFERENCE RECEIPT

Sending Department/Section _____

Date ____-_____-____ Responsible Name: _____ Initials _____

Time		DRN	- Ver	Receiving Dep/Sec	Responsible	Initials
____	____	_____	____	_____	_____	_____
____	____	_____	____	_____	_____	_____
____	____	_____	____	_____	_____	_____
____	____	_____	____	_____	_____	_____
____	____	_____	____	_____	_____	_____
____	____	_____	____	_____	_____	_____

Operator Name _____ Initials _____

Figure 23. Document Transference Receipt.

module into the system, was concerned with details and boundaries. The rationale for creating test cases was based in the "Equivalence Classes" and "Boundary-Value" testing techniques. Both are largely heuristic methods that provide the tester with a set of guidelines. The task was simplified because Oracle itself does most of the input validation. However, it is necessary to set the proper options using SQL*Forms to have those validations performed. Complex validation and integrity rules are entered by using embedded SQL statements inside the forms (triggers). To verify whether these operations were properly executed was a more difficult task. Some triggers update tables that are not presented on the form's screen. In this situation it is necessary to exit the testing module to access those tables. Some queries were tested before being

inserted into the form. These complex queries were developed using SQL*Plus. When the desired results were obtained, the query was integrated to the form with just a few modifications.

The data validation and integrity rules listed in Appendix C were largely used not only in developing the forms but also in testing them.

Table Locking

When an operator queries a block or updates a record in a block the base table for the block is automatically locked by SQL*Forms in "share update mode". This permits more than one operator to update the table concurrently. When a trigger acts in a table, SQL*Forms issues an "exclusive lock" on that table. If the table has already been locked in share update mode by another form, this will cause an error with the consequent roll back of the operations not committed yet. To avoid this situation a kind of a "two phase locking protocol" mixed with a "graph-based protocol" was used to "soften" the locking level. It is a two phase locking protocol in the sense that locks acquired are not released until a commit has been issue by the trigger or by the operator. It is a graph-based protocol because tables are accessed following an established partial ordering.

Table XIII shows the table ordering used in triggers to delete and update data that are not accessed from the block base table.

Implementation Difficulties

The most significant difficulty faced in the implementation phase came from a "bug" in the SQL*Form. The "Trigger Step Attribute Window" is a feature used when defining triggers. In this window it is possible

Table XIII. Partial Ordering for Locking Tables.

Table Name
Document
Reference
Annex
Doc_Log
Holder
Join
Microfilm
Subject
Classes
Terms
Notes
Relships
SeqNumbers
To_Delete

to define a success and/or a failure label to instruct the trigger about the next step to be executed after performing the actual trigger step. According the Oracle manual "SQL*Forms - Designer's Reference" (Zussman and others, 1987:8-29) we may define only the failure label, if desired. In this case when the trigger succeeds it is supposed to proceed to the next step and when it fails it is supposed to proceed to the failure label. Actually, if only the failure label was defined, the next time the form was loaded into the SQL*Forms, that label would not be saved into the INP file. This makes the trigger proceed to the next step on success or failure. There are some special conditions that avoids this behavior. If when loading that form you recall that trigger step and open that Trigger Step Attribute Window again, the label will be recorded back in the INP file. This "bug" causes confusing form behavior. The

form is tested and it is working well. Later, when some modification is introduced, the form starts to commit errors not related to the modification that was done. When the designer recalls the trigger to inspect the cause, he finds the single label there and no reason for the anomaly. But just because the trigger was recalled, the label is recorded back in the INP file and the form starts working well again. The developer becomes afraid of doing modifications in a form. It seems that whenever a form is loaded it starts making errors that stop only when the form is completely revised, even when nothing wrong is found. The solution is just to create a success label also, whenever a failure label becomes necessary. Success labels alone do not cause any problems.

VI. Conclusion

Although the final evaluation has to be made by the user, a natural way of measuring how well the system fulfills the requirements is by comparing the implementation to the system requirement specifications.

System Evaluation

A complete design and implementation of a complex database is a difficult and time consuming task. Even the requirements specification is a difficult task when the user does not know exactly what is needed, which frequently happens. The essence of the requirements are the necessity of a standard software to be run on PC like microcomputers, able to answer the questions presented in the Requirements Specification chapter, which are replicated in Table XIII for convenience. Because of this loose definition the evaluation is somewhat subjective. To show that the proposed system fulfills the needs the following paragraphs contrast the system with those questions.

Building the Database. The most important tasks in the system are those related to collecting, organizing, and updating the information about documents received and sent. Three of the four main modules in which the proposed system is divided are dedicated to these tasks. The fourth module, "Consult Database", concentrates the programs used to retrieve the stored information. In spite of their importance, most end-users will not have access to the maintenance modules. They will only use the module "Consult Database" to retrieve document information. Because of these system characteristics and because the system

Table XIV. Document Control Fundamental Questions.

a. Has the organization received this document?
b. What is this document about?
c. Where is this document?
d. Which sections/departments have already processed this document?
e. Who is currently analyzing this document?
f. Which documents may I destroy?
g. Which documents are related to this subject?

requirements are expressed as end-users' questions about documents, the capacity of responding well to those questions depends on having the right information stored in the database, and the flexibility in extracting information.

Answering the Questions. The goal of the maintenance modules is to keep a knowledge base that contains all information needed, logically divided into three partitions:

- Control Database
- Catalog Database
- Subject Database

The standard way of extracting this information is using any of the "Consult Database" modules. The basic Consult Database module is the "Retrieve Document". All document attributes are available in this module. In spite of that, some of the other consult-database modules may be more appropriate, depending on the type of search being performed. All consult-database modules use embedded SQL statements. If some

special situation arises, a specific SQL query may be entered, using SQL*Plus command line. This kind of query only requires of the user a better understanding of SQL statements. This flexibility offered by the SQL language enforces the necessity of having all relevant information about the document collection in the database.

The "control database" encompasses information about document destination, processing, validity, and custody. It has the information needed to answer questions "c", "d", "e", and "f". Besides the "Retrieve Document" module, answers for questions "c" and "e" may also be obtained using the module "Browse by Holder".

The "catalog database" encompasses information about document identification attributes entered into the system by the module "Register Document". It has the information needed to answer question "a" and question "b". One can query browse the database with identification arguments by using the module "Browse by Header".

Finally, the "subject database" encompasses all information needed to search for documents about a subject of interest. One can find the answer to question "g" by querying the database using thesaurus terms as arguments. This may be accomplished using the module "Browse by Subject".

Evaluation Conclusion.

By comparing the needs with the proposed system, one concludes the proposed system well satisfies the requirements. It satisfies the requirements because it has procedures easily implementable, to store all data necessary to respond the questions presented in the specification; because it provides standard and alternative ways to query about those

questions; and, finally, because it is appropriate for a standard software, since its procedures are based on official issues and it was implemented with a DBMS available for PC like microcomputers and several other computer hosts.

Extra Features. The system not only satisfies the requirements but also provides additional information and control. As an example we have the DRN Verifier, which was introduced to avoid incorrect updates by having the operator enter a different DRN than that of the document being updated. Another example of additional information and control is the "Show Statistics" module. It is impossible to detect instability in the system, to tell whether the document processing is becoming slower, or to take corrective actions if there are no standards to compare with. The referred module provides information for this kind of management. It gives the statistical mean and standard deviation of the document processing time, calculated for solved documents that where issued between two user selected dates. This information is based on the time lapse between the first and the last custody transference date-time recorded in the Doc_Log table (document history).

Another significant feature is the subject retrieval capability provided by the Thesaurus. Note that the thesaurus, because of its relative independence from the Catalog and Control databases, may be used for other applications in the same environment. The interface would be a table like the table Subject, that would relate thesaurus terms and the other system's primary key. In a school environment, for instance, could use a student register number as the primary key to retrieve information

related to the students or a course code to search for data about courses.

Recommendations

There are three main recommendations for future research. The first one is basic to permit the use of the proposed system's subject retrieval feature. The second is desirable to improve the system security. The third and last one is concerned with the next natural step in the proposed system upgrading.

Thesaurus Building. Defining the terms that will constitute the thesaurus is a very important and also difficult task. We recommend doing a initial list by collecting suggested terms already used in the organizations elected to have the system.

Passwords. Forms and reports in the proposed system use a master user name and password. The modules (forms) were organized and partitioned in such way that permit password modular assignments by operator. Operator responsibilities should be well defined and accesses must be granted based on these obligations.

Document Entering from a File. A natural upgrading is exchanging documents in computer file format. In this situation, an alternative way of entering document data in the system would be a program to parse the received document file and to insert automatically the document attribute values into the database. Oracle has a utility tool referred as Data Loader (ODL) that can be used to develop an automatic document entering module.

Conclusion

Overall, this work significantly contributes to solving a basic and important problem shared for numerous Brazilian Air Force organizations. The proposed system, as standard software, offers many possibilities, as to make it possible to query the databases of different organizations and to exchange documents in computer file format, using the existing telecommunication networks. It represents a large step ahead to accelerate the document processing and to improve the decision making.

Appendix A: Script File for Creation of Tables, Clusters, and Indexes

The following statements were used in the script file CREATE_T.SQL to create the database in the Oracles's environment. It was run by typing START CREATE_T from inside SQL*Plus and pressing Enter. The database may be moved or replicated using this file or using a export file (Oracle utility).

```
CREATE CLUSTER C_DOCUMENT
      (DRN          NUMBER(8));

CREATE       TABLE DOCUMENT
     (DRN          NUMBER(8)    NOT NULL,
      P_SC         NUMBER(2),
      P_OC         NUMBER(2),
      P_NUMBER     NUMBER(5),
      P_YEAR       NUMBER(2),
      IDNR         NUMBER(5)    NOT NULL,
      IDCOMPL      CHAR(13)     NOT NULL,
      ORGNAME      CHAR(20)     NOT NULL,
      ISSUEDT      DATE         NOT NULL,
      TITLE        CHAR(78)     NOT NULL,
      DOCTYPE      CHAR(7)      NOT NULL,
      CLASSIF      CHAR(1)      NOT NULL,
      VALIDITY     TIME         NOT NULL,
      JOINSTS      CHAR(1)      NOT NULL,
      PROCSTS      CHAR(1)      NOT NULL,
      FORMSTS      CHAR(1)      NOT NULL,
      HISTSTS      CHAR(1)      NOT NULL,
      ARCHSTS      CHAR(1)      NOT NULL)
CLUSTER      C_DOCUMENT (DRN);

CREATE       INDEX I_P_NUMBER
ON           DOCUMENT     (P_NUMBER ASC);

CREATE       INDEX I_IDNR
ON           DOCUMENT     (IDNR ASC);

CREATE       INDEX I_ORGNAME
ON           DOCUMENT     (ORGNAME);

CREATE TABLE REFERENCE
      (DRN          NUMBER(8)    NOT NULL,
```

```
      REFER         CHAR(20)     NOT NULL)
CLUSTER      C_DOCUMENT (DRN);

CREATE TABLE ANNEX
      (DRN          NUMBER(8)    NOT NULL,
      ANNEX         CHAR(20)     NOT NULL)
CLUSTER      C_DOCUMENT (DRN);

CREATE TABLE JOIN
      (MAIN         NUMBER(8)    NOT NULL,
      JOINED        NUMBER(8)    NOT NULL)
CLUSTER      C_DOCUMENT (MAIN);

CREATE TABLE MICROFILM
      (DRN          NUMBER(8)    NOT NULL,
      F_TYPE        CHAR(1)      NOT NULL,
      F_NO          NUMBER(4)    NOT NULL,
      FIRST         NUMBER(5)    NOT NULL,
      LAST          NUMBER(5))
CLUSTER      C_DOCUMENT (DRN);

CREATE TABLE DEPARTMENT
      (DEPCODE      CHAR(6)      NOT NULL);

CREATE TABLE HOLDER
      (DRN          NUMBER(8)    NOT NULL,
      DEPCODE       CHAR(6)      NOT NULL,
      NAME          CHAR(15))
CLUSTER      C_DOCUMENT (DRN);

CREATE TABLE DOC_LOG
      (DRN          NUMBER(8)    NOT NULL,
      LDATE         DATE         NOT NULL,
      LTIME         CHAR(5)      NOT NULL,
      SEND_DEP      CHAR(6)      NOT NULL,
      REC_DEP       CHAR(6)      NOT NULL)
CLUSTER      C_DOCUMENT (DRN);

CREATE VIEW DOC_LOG_VIEW AS
SELECT
TO_NUMBER (TO_CHAR (LDATE,'J')
           || SUBSTR(LTIME,1,2)
           || SUBSTR(LTIME,4,2))        JULIAN_D_T ,
DRN, LDATE, LTIME, SEND_DEP, REC_DEP
FROM DOC_LOG;

CREATE TABLE SUBJECT
      (DRN          NUMBER(8)    NOT NULL,
      TERM          CHAR(25)     NOT NULL)
CLUSTER      C_DOCUMENT (DRN);

CREATE       INDEX SUBJECT_TERM
```

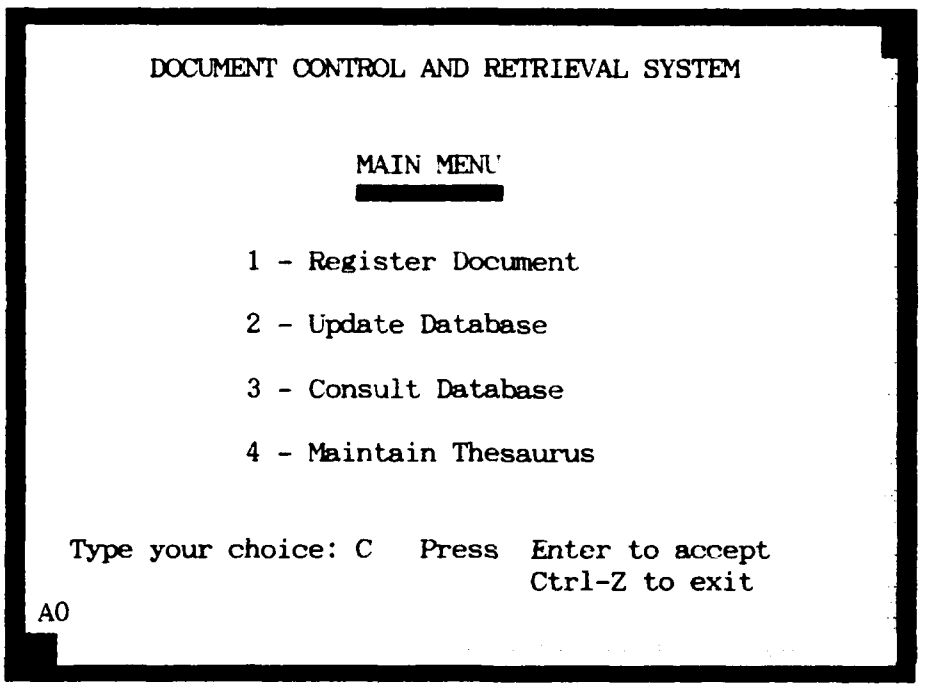
```
ON          SUBJECT          (TERM ASC);


CREATE CLUSTER CTERM
      (TERM       CHAR(25));

CREATE TABLE TERMS
      (TERM       CHAR(25)    NOT NULL,
      CLASSCD     CHAR(2)     NOT NULL)
CLUSTER     CTERM  (TERM);

CREATE TABLE CLASSES
      (CLASSCD    CHAR(2)     NOT NULL,
      CLASS       CHAR(20)    NOT NULL);

CREATE TABLE RELSHIPS
      (MAINTR     CHAR(25)    NOT NULL,
      REL         CHAR(2)     NOT NULL,
      SECTR       CHAR(25)
CLUSTER     CTERM  (MAINTR);

CREATE TABLE NOTES
      (TERM       CHAR(25)    NOT NULL,
      NOTE        CHAR(60)    NOT NULL)
CLUSTER     CTERM  (TERM);


CREATE TABLE DOC_TYPE
      (DOCTYPE    CHAR(7)     NOT NULL);

CREATE TABLE CLASSIFICATION
      (CLASSIF    CHAR(1)     NOT NULL,
      CLASSIF_NAME CHAR(12)   NOT NULL);

CREATE TABLE RELNAME
      (REL        CHAR(2)     NOT NULL,
      RNAME       CHAR(8)     NOT NULL);

CREATE TABLE SEQNUMBERS
      (LASTNUMBER   NUMBER(8),
      TABLENAME   CHAR(30));

CREATE TABLE TO_DELETE
      (DRN          NUMBER(8));


INSERT INTO CLASSIFICATION VALUES ('P','PUBLIC');
INSERT INTO CLASSIFICATION VALUES ('R','RESERVED');
INSERT INTO CLASSIFICATION VALUES ('C','CONFIDENTIAL');

INSERT INTO RELNAME VALUES ('BT','BROADER');
INSERT INTO RELNAME VALUES ('NT','NARROWER');
```

```
INSERT INTO RELNAME VALUES ('RT','RELATED');
INSERT INTO RELNAME VALUES ('US','USE');
INSERT INTO RELNAME VALUES ('UF','USED FOR');

INSERT INTO DEPARTMENT VALUES ('COMMAN');
INSERT INTO DEPARTMENT VALUES ('PROTOC');
INSERT INTO DEPARTMENT VALUES ('ARCH1');
INSERT INTO DEPARTMENT VALUES ('ARCH2');
INSERT INTO DEPARTMENT VALUES ('DEP1');
INSERT INTO DEPARTMENT VALUES ('DEP2');
INSERT INTO DEPARTMENT VALUES ('DEP3');
INSERT INTO DEPARTMENT VALUES ('DEP4');
INSERT INTO DEPARTMENT VALUES ('SEC1');
INSERT INTO DEPARTMENT VALUES ('SEC2');

INSERT INTO DOC_TYPE VALUES ('APRECIA');
INSERT INTO DOC_TYPE VALUES ('DESPACH');
INSERT INTO DOC_TYPE VALUES ('ENCAMIN');
INSERT INTO DOC_TYPE VALUES ('ESTIMAT');
INSERT INTO DOC_TYPE VALUES ('INFO');
INSERT INTO DOC_TYPE VALUES ('OF');
INSERT INTO DOC_TYPE VALUES ('RELAT');
INSERT INTO DOC_TYPE VALUES ('REL ESP');
INSERT INTO DOC_TYPE VALUES ('REL PER');
INSERT INTO DOC_TYPE VALUES ('REQ');
INSERT INTO DOC_TYPE VALUES ('BOL');

INSERT INTO SEQNUMBERS VALUES (89100000,'DOC_RECEIVED');
INSERT INTO SEQNUMBERS VALUES (89200000,'DOC_SENT');
INSERT INTO SEQNUMBERS VALUES (89100000,'PRINT_PS_RECEIVED');
INSERT INTO SEQNUMBERS VALUES (89200000,'PRINT_PS_SENT');

INSERT INTO CLASSES VALUES ('00','ROOT');
INSERT INTO CLASSES VALUES ('01','ORGANIZATION');

INSERT INTO TERMS VALUES ('ROOT','00');
INSERT INTO TERMS VALUES ('ORGANIZATION NAME','01');

INSERT INTO RELSHIPS VALUES ('ROOT','BT','');
INSERT INTO RELSHIPS VALUES ('ORGANIZATION','BT','ROOT');
INSERT INTO RELSHIPS VALUES ('ROOT','NT','ORGANIZATION');

COMMIT

EXIT
```

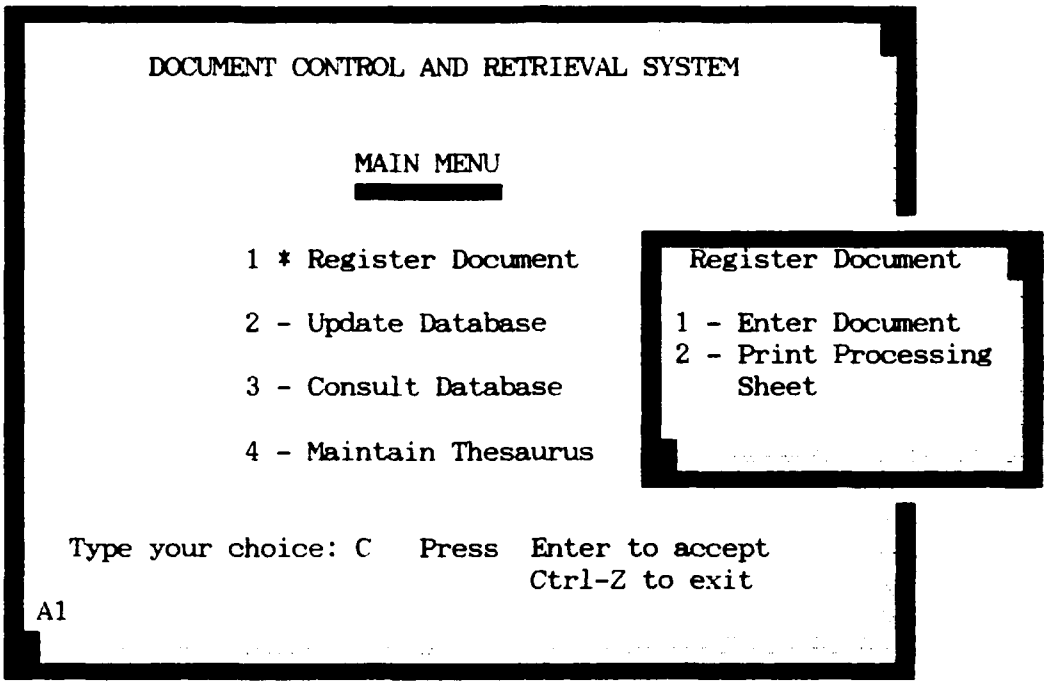
Appendix B: Screen Templates

The symbols and conventions used on the screen templates are listed in Table X.

Screen A0, Main Menu



Screen A1, Register Document



Screen A11, Enter Document

```
                    E N T E R   D O C U M E N T                Block 1

Automatic DRN C    DRN 99999999        Validity DD-MON-YY

  Protocol No 99-99/99999/99          Doc. Type CCCCCCC    Classif C

Doc. Identif. 99999/CCCCCCCCCCCCC    Issue Date DD-MON-YY

Organization CCCCCCCCCCCCCCCCCCCCCCCC  Status: C    C    C    C    C
Title                                                 Join Proc Form Hist Arch
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

            Reference   CCCCCCCCCCCCCCCCCCC                        Block 2
                                "
                                "

              Annex     CCCCCCCCCCCCCCCCCCC                        Block 3
                                "
                                "

A11                        F7 query    F10 insert/update    Ctrl-Z Menu
```

Screen A12, Print Processing Sheet

```
                    P R I N T   P R O C E S S I N G   S H E E T

        Printing Status:
                                          Received     Sent
        A - Last Document Printed         99999999     99999999

        B - Last Document Entered         99999999     99999999

          Options:

            1 - Print Received Documents from A to B.

            2 - Print Sent Documents from A to B.

            3 - Print a single document, given the DRN.

          Enter your choice: C

A12                                        Enter to print   Ctrl-Z to Menu
```

Screen A2, Update Database

```
        DOCUMENT CONTROL AND RETRIEVAL SYSTEM

                    MAIN MENU                          Update Database

                                                    1 - Generic Update
              1 - Register Document                 2 - Change Validity
                                                    3 - Doc. Destruction
              2 * Update Database                   4 - Join Documents
                                                    5 - Transfer Custody
              3 - Consult Database                  6 - Microfilm Data
                                                    7 - Subject Indexing
              4 - Maintain Thesaurus

    Type your choice: C    Press  Enter to accept
                                  Ctrl-Z to exit
A2
```

Screen A21-1, Generic Update

```
Page 1                  G E N E R I C    U P D A T E                Block 1

            DRN  99999999

  Protocol No 99-99/99999/99          Doc. Type CCCCCCC    Classif C

Doc. Identif. 99999/CCCCCCCCCCCCC     Issue Date DD-MON-YY

Organization CCCCCCCCCCCCCCCCCCCCCCCC  Status: C    C    C    C    C
Title                                              Join Proc Form Hist Arch
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

            Reference   CCCCCCCCCCCCCCCCCCC                        Block 2
                                "
                                "

              Annex     CCCCCCCCCCCCCCCCCCCC                       Block 3
                                "
                                "

A21-1    F7 query  F10 update  F6 del  PgDn next block/pg  Ctrl-Z Menu
```

Screen A21-2, Generic Update

```
Page 2                  G E N E R I C    U P D A T E

          HOLDER                                                   Block 4

        Department CCCCCCC    Holder Name CCCCCCCCCCCCCCC

          HISTORY                                                  Block 5

            Date       Time       Sender    Receiver
          DD-MON-YY   HH:MI       CCCCCCC   CCCCCCC
              "          "          "         "
              "          "          "         "
              "          "          "         "
              "          "          "         "
              "          "          "         "
              "          "          "         "
              "          "          "         "

          PgUp previous block/page          PgDn next block/page
A21-2     F7 query   F10 update/insert   Sh-F6 delete   Ctrl-Z to Menu
```

Screen A22, Change Validity

```
                    C H A N G E   V A L I D I T Y


      DRN 99999999  - 99            Validity  DD-MON-YY
            "         "                           "
            "         "                           "
            "         "                           "
            "         "                           "
            "         "                           "
            "         "                           "
            "         "                           "
            "         "                           "
            "         "                           "
            "         "                           "
            "         "                           "
            "         "                           "
            "         "                           "

A22                                F7 query   F10 update   Ctrl-Z Menu
```

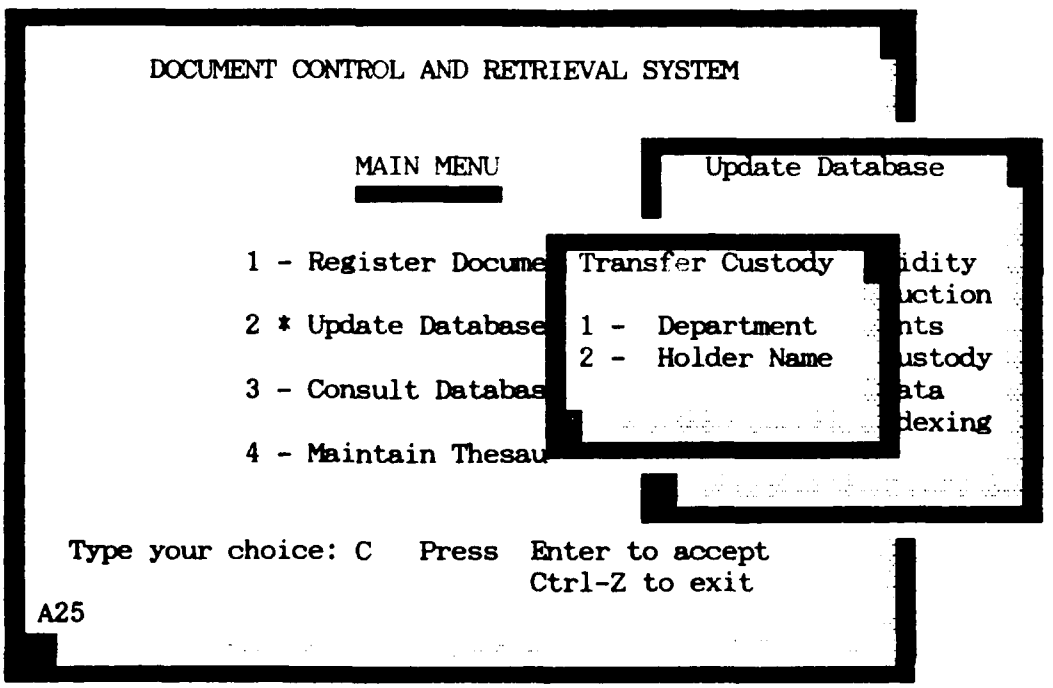
Screen A23, Document Destruction

```
                    D O C U M E N T   D E S T R U C T I O N

                                                                  Block 1
      Reference  CCCCCCCCCCCCCCCCCCCCCCCC

            DRN  99999999 - 99          Holder (CCCCCC)         Block 2
                   "         "                  "
                   "         "                  "
                   "         "                  "
                   "         "                  "
                   "         "                  "
                   "         "                  "
                   "         "                  "
                   "         "                  "
                   "         "                  "
                   "         "                  "
                   "         "                  "
                   "         "                  "

Note: Enter a reference to be inserted for each document processed
A23              F7 query   Shift-F6 process destruction   Ctrl-Z Menu
```

Screen A24, Join Documents

```
                    J O I N / S E P A R A T E   D O C U M E N T S


Main Document  99999999 - 99     Joined Document  99999999 - 99
                      "    "                           "    "
                      "    "                           "    "
                      "    "                           "    "
                      "    "                           "    "
                      "    "                           "    "
                      "    "                           "    "
                      "    "                           "    "
                      "    "                           "    "
                      "    "                           "    "
                      "    "                           "    "
                      "    "                           "    "
                      "    "                           "    "
                      "    "                           "    "

                                F3 duplicate field      F10 create/update joins
A24                             Shift-F6 separate       Ctrl-Z to Menu
```

Screen A25, Transfer Document Custody



Screen A251, Transfer Custody - Department

```
                T R A N S F E R   C U S T O D Y  -  DEPARTMENT

   Sending       Receiving
  Department    Department        Date        Time        Document
   9999999       9999999        DD-MON-YY    HH:MI      99999999-99
      "             "              "           "          "     "
      "             "              "           "          "     "
      "             "              "           "          "     "
      "             "              "           "          "     "
      "             "              "           "          "     "
      "             "              "           "          "     "
      "             "              "           "          "     "
      "             "              "           "          "     "
      "             "              "           "          "     "
      "             "              "           "          "     "
      "             "              "           "          "     "
      "             "              "           "          "     "

A251        F3 duplicate field     F10 create records     Ctrl-Z to Menu
```

Screen A252, Transfer Custody - Holder Name

```
                T R A N S F E R   C U S T O D Y  -  HOLDER NAME

      Department        Holder Name          Document
       CCCCCCC       CCCCCCCCCCCCCCC      99999999-99
          "                 "                "     "
          "                 "                "     "
          "                 "                "     "
          "                 "                "     "
          "                 "                "     "
          "                 "                "     "
          "                 "                "     "
          "                 "                "     "
          "                 "                "     "
          "                 "                "     "
          "                 "                "     "
          "                 "                "     "

A252                                  F10 update      Ctrl-Z Menu
```

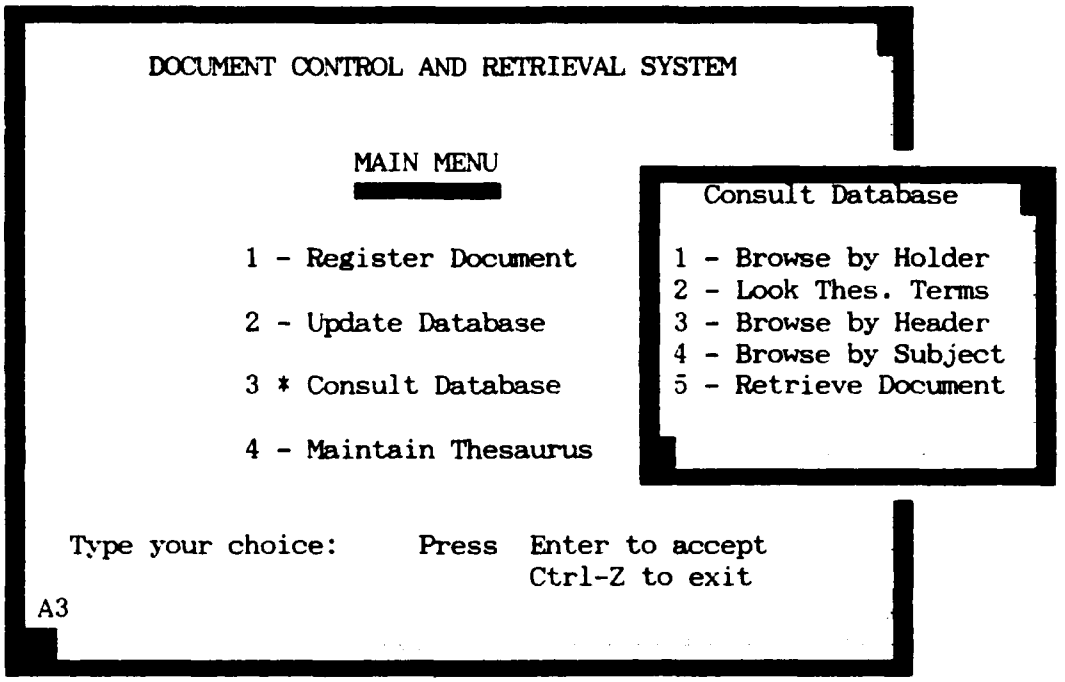
Screen A26, Update Microfilm Data

```
                    U P D A T E   M I C R O F I L M   D A T A

        DRN          Ver     Type    Number    First Frame    Last Frame
    99999999  -      99       C       9999        99999          99999
       "      -      "        "        "           "              "
       "      -      "        "        "           "              "
       "      -      "        "        "           "              "
       "      -      "        "        "           "              "
       "      -      "        "        "           "              "
       "      -      "        "        "           "              "
       "      -      "        "        "           "              "
       "      -      "        "        "           "              "
       "      -      "        "        "           "              "
       "      -      "        "        "           "              "
       "      -      "        "        "           "              "
       "      -      "        "        "           "              "
       "      -      "        "        "           "              "

A26    F7  query   F10 insert/update     Shift-F6 delete     Ctrl-Z Menu
```

Screen A27, Subject Indexing

```
                        S U B J E C T   I N D E X I N G

      DRN          Ver          Thesaurus Term                Class Name

   99999999  -     99     CCCCCCCCCCCCCCCCCCCCCCCCC     CCCCCCCCCCCCCCCCCCC
      "          "                   "                          "
      "          "                   "                          "
      "          "                   "                          "
      "          "                   "                          "
      "          "                   "                          "
      "          "                   "                          "
      "          "                   "                          "
      "          "                   "                          "
      "          "                   "                          "

                F9   look up terms                 Shift-F6  delete
A27             F10  insert/update                  Ctrl-Z    Menu
```

Screen A3, Consult Database



Screen A31, Browse Documents By Holder

```
                    B R O W S E   B Y   H O L D E R

          ->                                    ->     ->  Date of     ->
Depart Holder N. DRN      Prot#/yy Doc Ident.  Type Register  Title
CCCCCC CCCCCCCC 99999999 99999 99 99999 CCCCC CCCC DD-MON-YY CCCCCCCCCC
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
  "      "        "       "     "  "     "     "    "        "
A31   F7 query  :V variable  % wildcard(*)  _ wildcard(?)  Ctrl-Z Menu
```

Screen A32, Look Up Thesaurus Terms

```
        DOCUMENT CONTROL AND RETRIEVAL SYSTEM

                     MAIN MENU
                                       Consult Database
               1 - Register Docume  Look Up Thes. Terms   r
                                                          s
               2 - Update Database  1 - Classes           r
                                    2 - Terms & Relations ct
               3 * Consult Databas  3 - Terms within Class nt
                                    4 - Terms Sequentially
               4 - Maintain Thesau

  Type your choice:     Press  Enter to accept
                               Ctrl-Z to exit
 A32
```

Screen A321, Look Up Thesaurus Classes

```
             L O O K   U P   T H E S A U R U S - CLASSES

             Class Code  CC        Name  CCCCCCCCCCCCCCCCCCCCC
                         "                      "
                         "                      "
                         "                      "
                         "                      "
                         "                      "
                         "                      "
                         "                      "
                         "                      "
                         "                      "
                         "                      "
                         "                      "
                         "                      "
                         "                      "
                         "                      "
                         "                      "
                         "                      "

A321   F7 query  :V variable  % wildcard(*)  _ wildcard(?)  Ctrl-Z Menu
```

Screen A322, Look Up Terms and Relationships

```
     L O O K   U P   T H E S A U R U S - TERMS AND RELATIONSHIPS

     Term                    Class Code          Class Name          Block 1

CCCCCCCCCCCCCCCCCCCCCCCCCC   CC          (CCCCCCCCCCCCCCCCCCCC)

                  (CCCCCCCC)      (CCCCCCCCCCCCCCCCCCCCCCCCCC)      Block 2
                      "                       "
                      "                       "
                      "                       "
                      "                       "
                      "                       "
                      "                       "
                      "                       "
                      "                       "
                      "                       "
                      "                       "
                      "                       "

A322    F7 query   F5 copy   V: variable   %  _ wildcards   Ctrl-Z Menu
```

Screen A323, Look Up Terms Within Classes

```
       L O O K   U P   T H E S A U R U S - TERMS WITHIN CLASSES

     Class Code                          Term
        CC  CCCCCCCCCCCCCCCCCCCC      CCCCCCCCCCCCCCCCCCCCCCCCCC
         "              "                        "
         "              "                        "
         "              "                        "
         "              "                        "
         "              "                        "
         "              "                        "
         "              "                        "
         "              "                        "
         "              "                        "
         "              "                        "
         "              "                        "
         "              "                        "
         "              "                        "
         "              "                        "
         "              "                        "
         "              "                        "

A323   F7 query  :V variable  % wildcard(*)  _ wildcard(?)  Ctrl-Z Menu
```

Screen A324, Look Up Terms in Sequential Order

```
    L O O K   U P   T H E S A U R U S - TERMS IN SEQUENTIAL ORDER

            Name                          Class Code
      CCCCCCCCCCCCCCCCCCCCCCCCC          CC  (CCCCCCCCCCCCCCCCCCC)
                        "                   "              "
                        "                   "              "
                        "                   "              "
                        "                   "              "
                        "                   "              "
                        "                   "              "
                        "                   "              "
                        "                   "              "
                        "                   "              "
                        "                   "              "
                        "                   "              "
                        "                   "              "
                        "                   "              "
                        "                   "              "
                        "                   "              "

A324   F7 query  :V variable  % wildcard(*)  _ wildcard(?)  Ctrl-Z Menu
```

Screen A33, Browse Documents by Headers

```
                          B R O W S E   B Y   H E A D E R S
                        ->          ->      ->                      ->
  DRN       ID#    Issue Org.   Type C Issue.Dt                Title

99999999 CCCCC CCCCCCCCCCCC CCCC C DD-MON-YY CCCCCCCCCCCCCCCCCCCCCCCCC
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "
   "       "       "          "    "      "                      "

A33   F7 query  :V variable  % wildcard(*)  _ wildcard(?)  Ctrl-Z Menu
```

Screen A34-1, Browse Documents By Subject

```
Page 1                B R O W S E   B Y   S U B J E C T

 Term to browse by          Other terms for which the document is indexed
CCCCCCCCCCCCCCCCCCCCCCCCC (CCCCCCCCCCCCCCCCCCCC CCCCCCCCCCCCCCCCCCCC)
            "                        "                    "
            "                        "                    "
            "                        "                    "
            "                        "                    "
            "                        "                    "
            "                        "                    "
            "                        "                    "
            "                        "                    "
            "                        "                    "
            "                        "                    "
            "                        "                    "
            "                        "                    "
            "                        "                    "
            "                        "                    "
            "                        "                    "
            "                        "                    "

A34-1                       F7 query      Tab next page      Ctrl-Z Menu
```

Screen A34-2, Browse Documents By Subject

```
Page 2                  B R O W S E   B Y   S U B J E C T
                      ->        ->        ->                  ->
  DRN      ID#    Issue Organ Type C Issue Dt                Title

99999999 99999 CCCCCCCCCCCC CCCC C DD-MON CCCCCCCCCCCCCCCCCCCCCCCCCCCC
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "
   "        "         "        "   "   "                      "

A34-2                       F7 query      Tab previous page      Ctrl-Z Menu
```

Screen A35-1, Retrieve Documents

```
Page 1      R E T R I E V E   D O C U M E N T S  (COMPLETE)    Block 1

          DRN 99999999 99              Validity DD-MON-YY

  Protocol No 99-99 / 99999 / 99          Doc. Type CCCCCC  Classif C

Doc. Identif. 99999 / CCCCCCCCCCCC      Issue Date DD-MON-YY

Organization CCCCCCCCCCCCCCCCC  Status: C      C     C    C    C
Title                                        Join Process Form Hist Arch
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

Subject Index:                                                  Block 2
DRN 99999999  Term CCCCCCCCCCCCCCCCCCCCCCC Class CCCCCCCCCCCCCCCCCC
       "        "              "                          "
       "        "              "                          "
       "        "              "                          "
       "        "              "                          "
       "        "              "                          "

A35-1     F6 help   F7 query   F5 browse   PgDn see more   Ctrl-Z Menu
```

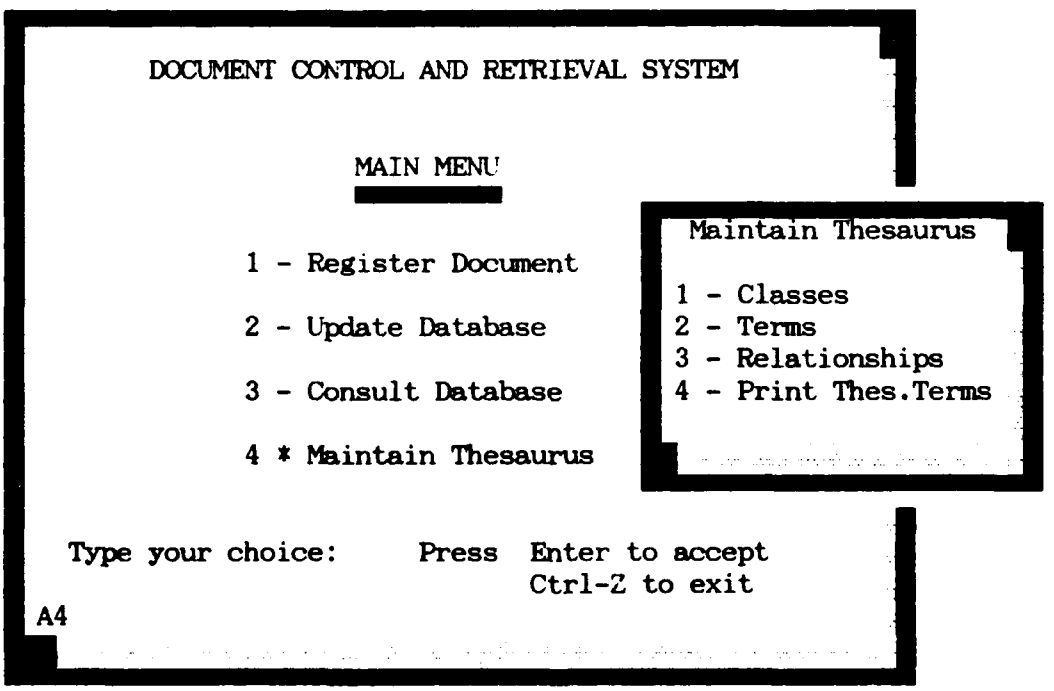
Screen A35-2, Retrieve Documents

```
Page 2           R E T R I E V E   D O C U M E N T S

Reference:  CCCCCCCCCCCCCCCCCCC  (999999999)                  block 3

    Annex:  CCCCCCCCCCCCCCCCCCC  (999999999)                  block 4

     Join:  Main  99999999    Joined  99999999                block 5

Microfilm: Type C  # 9999  First 99999  Last 99999 (99999999) block 6

   Holder:  Name CCCCCCCCCCCCCCC   Dep CCCCCC   DRN(99999999) block 7

History:  Date DD-MON-YY HH:MI  Send CCCCCC  Rec CCCCCC  DRN (9999999)
                "        "        "          "           "
 block 8        "        "        "          "           "
                "        "        "          "           "
                "        "        "          "           "
                "        "        "          "           "
                "        "        "          "           "

A35-2                          F7 query    PgUp page 1    Ctrl-Z Menu
```

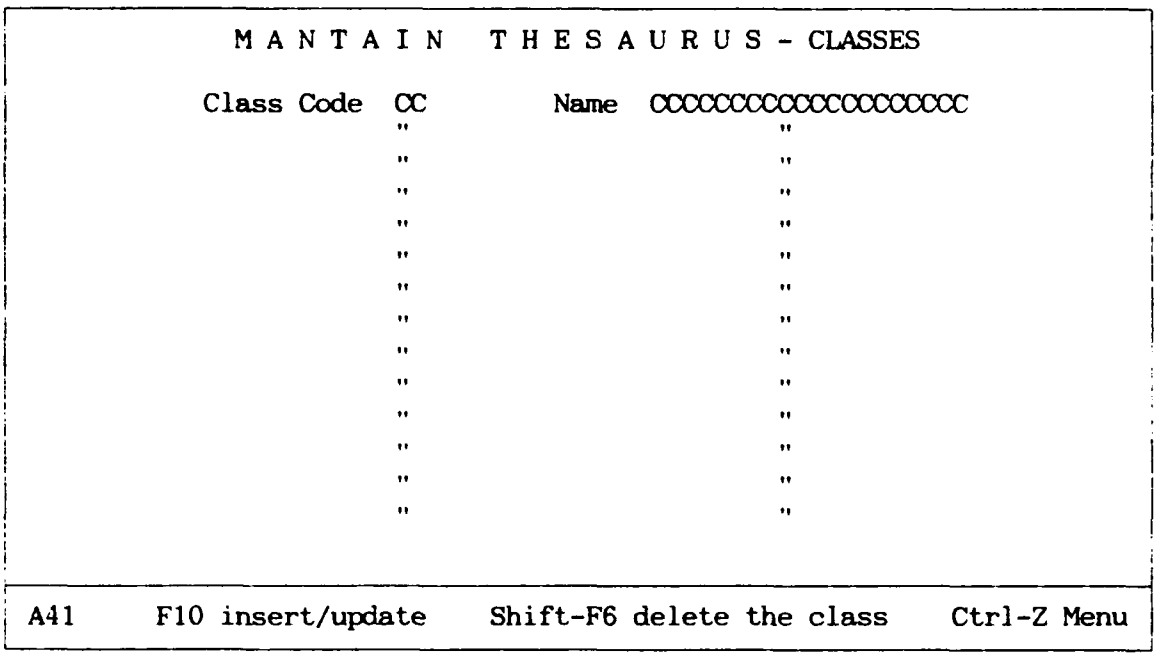
Screen A36, Show Statistics

```
                    S H O W    S T A T I S T I C S

  Enter period:  from DD-MON-YY to DD-MON-YY

 Total # doc 999999   Processing time in days: mean 999   std dev 9999
Status:                                                      Processing
            Form   %                 Join   %                    Phase   %
     paper 999999 999       indep  999999 999       processing 999999 999
 microfilm 999999 999       main   999999 999          waiting 999999 999
 destroyed 999999 999       joined 999999 999           solved 999999 999

           History %                                        Archive Sec %
    normal 999999 999                                  classified 999999 999
alteration 999999 999                                unclassified 999999 999

Thesaurus:
 Classes 99   terms 999999    Scope Notes 999999    Relationships 999999

Indexed doc: # 999999 % 999   # of terms by indexed doc: avrg 99 std 999

 A36                    F5 calculate      F7 clean fields      Ctrl-Z Menu
```

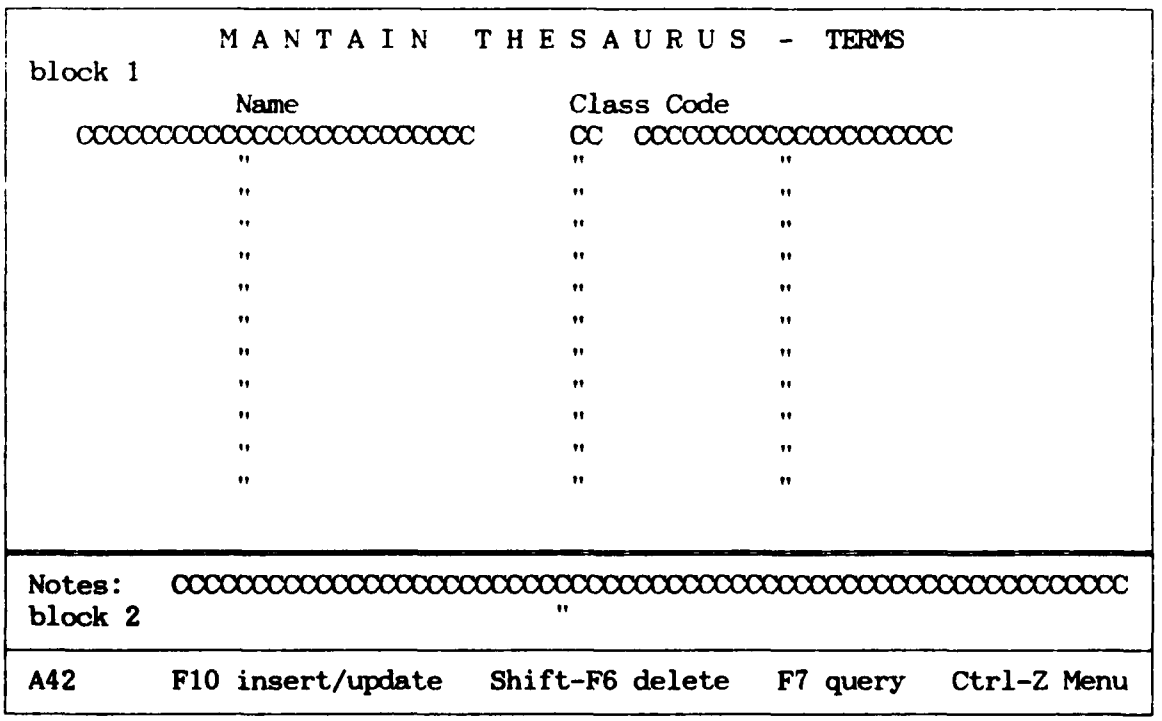
Screen A4, Maintain Thesaurus



Screen A41, Maintain Thesaurus Classes



Screen A42, Maintain Thesaurus Terms



Screen A43, Maintain Relationships

```
                M A N T A I N   T H E S A U R U S  -  RELATIONSHIPS

        Main Term                    Relation          Secondary Term
CCCCCCCCCCCCCCCCCCCCCCCCCCC  CC CCCCCCCC  CCCCCCCCCCCCCCCCCCCCCCCCCCC
                             "   "                    "
                             "   "        "           "
                             "   "        "           "
                             "   "        "           "
                             "   "        "           "
                             "   "        "           "
                             "   "        "           "
                             "   "        "           "
                             "   "        "           "
                             "   "        "           "
                             "   "        "           "
                             "   "        "           "
                             "   "        "           "
                             "   "        "           "

A43     F10 insert/updt    Shift-F6 del    F5 look up terms    Ctrl-Z Menu
```

Screen A44, Print Thesaurus Terms

```
                M A N T A I N   T H E S A U R U S  -  PRINT TERMS


                      1 - ORDER BY CLASS, TERM

                      2 - ORDER BY TERM

                      3 - ORDER BY TERM HIERARCHICAL ASSOCIATION

                      4 - ORDER BY TERM, GIVING ALL RELATIONSHIPS


                    Choice C


A44                                        Enter accept choice    Ctrl-Z Menu
```

Appendix C: Validation and Integrity Rules

Modules A0, A1, A2, A25, A3, A32, A4 - Menu Tree

Input Condition: Insertion, Update, Deletion, and Query.

. Do not permit these operations in the module.

Input Condition: Value of Choice.

. According the value of the choice call the appropriated module.

Module A11 - Enter Document

Input Condition: Insertion.

. Verify the primary key of the table Document: DRN.

. Verify the primary key of the table Reference: DRN, REFER.

. Verify the primary key of the table Annex: DRN, ANNEX.

. Generate automatically the DRN for a received document when the control field "Automatic DRN" has the value R.

. Generate automatically the DRN for a sent document when the control field "Automatic DRN" has the value S.

. Permit the operator to type the DRN that will be used for the insertion when the control field "Automatic DRN" has the value N (none).

. Skip the DRN field when the control field is set to generate the DRN automatically.

. Insert automatically a validity date 5 years ahead of the insertion date.

. Verify if exists in the table Doc_Type the value entered in the document type field.

. Display by operator request, in the field Document Type, the values available for this field.

. Verify whether there exists in the table OrgName the value entered in the field Organization.

. Verify whether there exists in the table Classification the value entered in the field Classification.

. Display by operator request, in the field Document Classification, the values available for this field.

. Insert automatically the following status values:
  . Join = I,
  . Process = P,
  . Form = P,
  . History = N.

. Insert automatically the Archive status:

. If document classification is P (public), then ARCHSTS = U;

. If document classification is R or C, then ARCHSTS = C.

. Insert automatically a row in the DOC_LOG table with the following values:
  . DRN = the document's DRN,
  . LDATE and LTIME = date and time of registering,
  . SEND_DEP = PROTOC,
  . REC_DEP = COMMAN.

. Insert automatically a row in the HOLDER table with the following values:
  . DRN = document DRN,
  . DEPCODE = COMMAN,
  . NAME = NULL.

Input Condition: Deletion.

. Do not permit to delete a document (block 1).

Input Condition: Update.

. Do not permit to change the DRN, Validity, and all status fields.

Input Condition: Query.

. Order queries by DRN in descendent order.

. Permit to enter a query condition based on document table values.

. Display automatically, when the operator queries the document block, the Reference and Annex rows that correspond to the DRN in the DRN field.

Module A12 - Print Processing Sheet

Input Condition: Pre-Form.

. Show the DRN of the last document entered and the DRN of the last document printed, either received and sent.

Input Condition: Value of Choice.

. When Choice is 1, generate the processing sheets that correspond to the received documents that have a DRN higher than the DRN of the last document printed, and a DRN smaller or equal the DRN of the last document entered.

. When Choice is 2, generate the processing sheets that correspond to the sent documents that have a DRN higher than the DRN of the last document printed, and a DRN smaller or equal the DRN of the last document entered.

. When Choice is 3, ask for a DRN to print the correspondent processing sheet.

. In either choices 1, 2, and 3, also direct the output to a DOS ASCII file.

. After printing choices 1 and 2, update the table Seqnumbers (where the last DRN entered and printed are kept) setting the last DRN printed equal the last DRN entered.

Module A21 - Generic Update

Input Condition: Insertion.

. Do not permit insertion in the table Document (to create a new document).

. Insertions in all other tables must use the same DRN of table Document.

. Verify the primary key of the table Reference: DRN, REFER.

. Verify the primary key of the table Annex: DRN, ANNEX.

. Verify the primary key of the table Holder: DRN.

. Verify the primary key of the table Doc_Log: DRN, LDATE, SEND_DEP.

. Rules for Holder and Document Log tables are the same of those on update input condition.

Input Condition: Update.

. Do not permit to update the DRN in the table Document.

. Verify in the table Doc_Type the existence of the value entered in the field Document Type.

. Verify in the table OrgName the existence of the value entered in the field Organization.

. Verify in the table Classification the existence of the value entered in the field Classification.

. If the document is Joined or Destroyed, do not permit to access the Holder and the Document Log tables.

. Update the table Holder automatically when a row is inserted, updated, or deleted in the table Doc_Log (Historic). Set the field Holder Name equal null and the field Holder Department equal the latest department that received the document.

. Set the document history status equal "A" (alteration) if there exists a row in the table Doc_Log in which the sending department is different from the receiving department in the previous row. Otherwise, set the document history status equal "N" (normal). Example: assume there are two rows in the Doc_Log table, ordered by DRN, Date, and Time, in ascendent order, with these values:

DRN	Date	Time	Send_Dep	Rec_Dep
99999999	DATE1	TIME1	DEP1	DEP2
SAME DRN	DATE2	TIME2	DEP3	DEP4

The underlined departments should be the same. As they are not the same, set the document history status equal A.

Input Condition: Deletion.

. If the document to be deleted is a Main document, delete also all corresponding joined documents in the join table.

. Each document deleted from the table document, has to be deleted also from the following tables:

- . Reference
- . Annex
- . Doc_Log
- . Holder
- . Join
- . Microfilm
- . Subject

Input Condition: Query.

. Order queries by DRN in descendent order.

. Permit to enter a query with conditions based only on columns of table Document. Query all other tables by the DRN entered in the table Document.

. Display automatically, when the operator queries the document block, the Reference and Annex rows that correspond to the document.

. If the document is Joined or Destroyed, do not permit to access the Holder and the Document Log tables.

. Display automatically, when the operator turns to the second page of the form, the Holder and Doc_Log rows that corresponds to the DRN entered on table Document.

. Display again the rows of tables Doc_Log and Holder after an updating, deletion or insertion in table Doc_Log.

Module A22 - Change Validity

Input Condition: Insertion and Deletion.

. Do not permit these operations in the module.

Input Condition: Update.

. Before updating, test the Verifier correctness.

Form A23 - Process Document Destruction

Input Condition: Insertion and Update.

. Do not permit these operations in the module.

Input Condition: Process Document Destruction.

. Test the Verifier(s) correctness.

. If the document is a Main document in a Join relationship process also the document that are joined to it.

. Delete the document(s) from the tables:

- . Reference
- . Annex
- . Doc_Log
- . Holder
- . Join
- . Microfilm
- . Subject

. Set the status of each document equal "D" (table Document).

. Insert a reference in the Reference table for each document that was processed.

Module A24 - Join Documents

Input Condition: Update.

. Do not permit this operation in the module.

Input Condition: Insertion.

. Verify the primary key of the table Join: JOINED.

. Verify whether document exists and its form status is not destroyed.

. Verify whether the Main document is already a Joined document.

. Verify whether the Joined document is already in the table.

. Verify whether the Main and Joined documents are the same.

. Test Verifiers correctness.

. Set the join status of the Main document equal "M", and of the Joined document equal "J".

. Delete the Historic and Holder of the joined document.

Input Condition: Deletion.

. Set the join status of the Joined document equal "I".

. Set the joined document holder department and holder name equal to those of the Main document.

. Insert in the joined document log (History) the latest transference in the main document history.

. Set the join status of the Main document equal "I", only if there is not any other document joined to it.

Module A251 - Transfer Custody Department

Input Condition: Update and Deletion.

. Do not permit these operations in this form.

Input Condition: Insertion.

. Verify the primary key of the table Doc_Log: DRN, LDATE, SEND_DEP.

. Verify the existence of the sending and receiving departments.

. Verify the existence of the document in the Doc_Log table.

. Verify whether the document is joined.

. Verify whether the document was destroyed.

. Test Verifier correctness.

. Update the table Holder. Set the field Holder Name equal null and the field Holder Department equal the receiving department of the latest row.

. Set the document history status equal "A" (alteration) if there exists a transference of the document in which the sending department is different from the receiving department in the previous row. Otherwise, set the document history status equal "N" (normal). Example: assume there are two rows in the Doc_Log table, ordered by DRN, Date, and Time, in ascendent order, with these values:

DRN	Date	Time	Send_Dep	Rec_Dep
99999999	DATE1	TIME1	DEP1	DEP2
SAME DRN	DATE2	TIME2	DEP3	DEP4

The underlined departments should be the same. As they are not the same, set the document history status equal A.

Module A252 - Transfer Custody Holder Name

Input Condition: Insertion and Deletion.

. Verify the primary key of the table Holder: DRN.

. Do not permit these operations in the module.

Input Condition: Update.

. Update only the Holder Name.

. Test the DRN Verifier before updating.

Input Condition: Query.

. Query by all columns of table Holder.

Module A26 - Update Microfilm Data

Input Condition: Insertion.

. Verify the primary key of the table Microfilm: DRN, F_TYPE, F_NO, FIRST.

. Verify whether the document exists in table Document.

. Test Verifier correctness.

. Change Form status to M (microfilmed).

Input Condition: Update and Deletion.

. Test Verifier Correctness.

. Do not permit to update the DRN field.

. Update Form status to P (paper) when the last row of a document is deleted from the table Microfilm.

Module A27 - Subject Indexing

Input Condition: Insertion and Update.

. Verify the primary key of the table Subject: DRN, TERM.

. Verify the document existence in the table document.

. Test the Verifier correctness.

Input Condition: Deletion.

. Test the Verifier correctness.

Input Condition: Query.

. Display the class of each term.

. Query by all columns of table Subject.

. Permit a direct access to Look up Terms e Relationships form.

. Permit to copy a term from the Look up Terms form and paste it on the Subject Indexing form.

Module A31 - Browse by Holder

Input Condition: Insertion, Update, and Deletion.

. Do not permit these operations in the module.

Input Condition: Query.

. Query by the holder department, holder name, and DRN.

. Order queries by department, holder name, and DRN in descendent order.

. For each document retrieved by the query, show the corresponding columns of table Document.

Module A321 - Look up Thesaurus Classes

Input Condition: Insertion, Update, and Deletion.

. Do not permit these operations in the module.

Input Condition: Query.

. Order queries by CLASS, CLASSCD (class code).

Module A322 - Look up Thesaurus Terms and Relations

Input Condition: Insertion, Update, and Deletion.

. Do not permit these operations in the module.

Input Condition: Query.

. Query by term and class.

. Order queries by CLASSCD (class code), TERM.

. For each term retrieved by the query, show the existing scope note, from the table Notes, and the existing relationships and corresponding terms, from the table Relships.

. Permit to copy a term into a memory variable to be possible to paste the term in a calling module.

Module A323 - Look up Thesaurus Terms within Classes

Input Condition: Insertion, Update, and Deletion.

. Do not permit these operations in the module.

Input Condition: Query.

. Query by term and class.

. Order queries by CLASSCD, TERM.

Module A324 - Look up Thesaurus Terms Sequentially

Input Condition: Insertion, Update, and Deletion.

. Do not permit these operations in the module.

Input Condition: Query.

. Query by term.

. Order queries by TERM.

. For each term retrieved, show the corresponding class name, from the table Classes.

Module A33 - Browse by Header

Input Condition: Insertion, Update, and Deletion.

. Do not permit these operations in the module.

Input Condition: Query.

. Query by DRN, document identification number, issuing organization, issued date, document type, classification, and document title.

. Order queries by DRN in descendent order.

Module A34 - Browse by Subject

Input Condition: Insertion, Update, and Deletion.

. Do not permit these operations in the module.

Input Condition: Query.

. Query by term, from the subject indexing table.

. For each document retrieved, show other terms under which the document is also indexed.

. For each document retrieved, show from the table Document its corresponding DRN, document identification number, issuing organization, issued date, document type, classification, and document title.

. Order queries by term and DRN in descendent order.

Module A35 - Retrieve Documents

Input Condition: Insertion, Update, and Deletion.

. Do not permit these operations in the module.

Input Condition: Query.

. Query by all attributes a document may have, from all tables.

. Order queries by DRN in descendent order.

Module A36 - Show Statistics

Input Condition: Insertion, Update, and Deletion.

. Do not permit these operations in the module.

Input Condition: Query.

. Query tables by a period of time.

. Calculate statistics from database tables.

Module A41 - Maintain Thesaurus Classes

A class is also a thesaurus term. We may think in a class as a implicit one level hierarchical relationship between a term (the class name) and other terms which are subordinated to it. Explicit hierarchical relationships are expressed by the hierarchical relationship. All terms that have explicit hierarchical relationships have to be descendants of the term ROOT, which is the root of the

explicit hierarchical relationship tree. When a class is created it is also created a hierarchical relationship connecting the class name (actually the term which corresponds to a class name) to the root of the hierarchical tree. All hierarchical relationships subtrees of this class that will be further created will be connected to the hierarchical tree trough the class name.

Because of these characteristics, some operations in the table Class have consequences in the tables Terms, Notes, Subject, and Relationships.

Input Condition: Insertion.

. Verify the primary key of table Classes: CLASSCD.

. Verify the uniqueness of the class name in the table Classes.

. Verify the uniqueness of the class name in the table Terms.

. Insert automatically in the table TERMS a term corresponding to the class being created (which will be referred as the "class-term" like the example below:

```
Table Classes: Class Name = 'JOBS',  Class Code = '04'
Table Terms:    Term Name = 'JOBS',  Class Code = '04'
```

. Insert automatically in the table Notes a note equal "CLASS" corresponding to the class-term.

. Insert automatically in the table Relships a hierarchical relationship (and the correspondent reciprocal entry) where the main term is the class-term that is being inserted, the relationship is broader term, and the secondary term is "ROOT".

Input Condition: Update.

. Verify the primary key of table Classes: CLASSCD.

. Verify the uniqueness of the class name in the table Classes.

. Verify the uniqueness of the class name in the table Terms.

. Do not permit updating of classes 00 and 01. Give a message saying they are owned by the system.

. Update in the table Subject the corresponding terms.

. Update in the table Terms the corresponding term.

. When the class code was updated, update in table Terms the class code of all terms that pertain to the class being update .

. Update in table Notes the corresponding terms.

. Update in table Relships the rows that have either the main term or the secondary term equal the class name.

Input Condition: Deletion.

. Do not permit deleting the classes 00 and 01. Give a message saying they are owned by the system.

. Delete in the table Subject all rows that have the column term equal the class being deleted.

. Delete in the table Relships the rows that have either the main term or the secondary term equal a term pertaining to the class being deleted.

. Delete in the table Notes the rows that have the column term equal a term pertaining to the class being deleted.

. Delete in the table Terms the rows that have the column class code equal the class code of the class being deleted.

Input Condition: Query.

. Order by class code.

. Query by class code and class name.

Module A42 - Maintain Thesaurus Terms

Input Condition: Insertion.

. Verify the primary key of the table Terms: TERM.

. Verify the primary key of the table Notes: TERM, NOTE.

Input Condition: Update.

. Do not permit updating class-terms.

. Update in the table Subject the corresponding terms.

. Update in table Notes the corresponding terms.

. Update in table Relships the rows that have either the main term or the secondary term equal the class name.

Input Condition: Deletion.

. Delete in the table Subject all rows that have the column term equal the term being deleted.

. If there exist hierarchical relationships in the table Relships involving the term being deleted, disconnect the deleting term from the hierarchical tree.

. Delete in the table Relships the rows that have either the main term or the secondary term equal the term being deleted.

. Delete in the table Notes the rows that have the column term equal the term being deleted.

Input Condition: Query.

. Order queries by TERM (table Terms).

. Display for each term shown in the table Terms the corresponding class name.

. Display for each term shown in the table Terms the corresponding scope notes.

. When querying the table Scope Notes, display for each scope note the correspondent terms.

. Permit to copy and paste between blocks and to and from other forms.

Module A43 - Maintain Thesaurus Relationships

The Hierarchical Tree.

. The hierarchical relationships form a hierarchical tree. The root of this tree is the term "root", which is the single term of a class created by the system, also named "root", which has the class code "00". The nodes at the first level of this tree are composed of class names. Therefore, each class is a child of the root. These relationships are maintained automatically when a class is created, updated or deleted. All terms that have hierarchical relationships are descendants of their classes.

Input Condition: Insertion.

. Do not permit the operator to insert relationships having "root" as the main term or secondary term.

. Verify the primary key of the table Relships: main term, relationship, and secondary term.

. Verify the term existence in the table Terms.

. Verify the uniqueness of a main term in a "use" relationship, and its reciprocal entry secondary term in a "used for" relationship.

. Verify uniqueness of MAINTR, 'BT' (broader term) relationship and its reciprocal entry SECTR, 'NT' (narrower term).

. Do not create hierarchical relationships with terms that have a "use" relationship.

. Do not permit a term to have both the "use" and the "used for" synonym relationships.

. Verify that both terms in a "broader term" or in a "narrower term" relationship pertain to the same class.

. Connect automatically to the class in the hierarchical relationship tree, hierarchical relationships between terms that are disconnected from the tree. As an example suppose "microcomputer" and "keyboard" are terms pertaining to the class "computer" and that no previous hierarchical relationships were created involving these two terms. The insertion of the relationship

<microcomputer, NT, keyboard>

will generate automatically the following relationships:

<keyboard, BT, microcomputer>,	which is the reciprocal entry,
<microcomputer, BT, computer>	to connect the previous relation to the hierarchical tree,
<computer, NT, microcomputer>	which is the reciprocal entry.

. Insert the reciprocal of the hierarchical (broader/narrower) relationship automatically.

. Insert the reciprocal of the associative (related) relationship automatically.

. Insert the reciprocal of the synonym (use/used for) relationship automatically.

. Create automatically a hierarchical relationship between a term and its class name when the broader term, in a hierarchical relationship being inserted, is disconnected from the hierarchical tree in the table RELSHIPS.

Input Condition: Update.

. Permit to update only the main term.

. Do not permit the operator to update relationships having "root" as the main or secondary term.

. Verify that both terms in a "broader term" or in a "narrower term" relationship pertain to the same class.

. Update the reciprocal of the hierarchical (broader/narrower) relationship automatically.

. Update the reciprocal of the associative (related) relationship automatically.

. Update the reciprocal of the synonym (use/used for) relationship automatically.

Input Condition: Deletion.

. Do not permit the operator to delete relationships having "root" as the main or secondary term.

. Delete the reciprocal of the hierarchical (broader/narrower) relationship automatically.

. Delete the reciprocal of the associative (related) relationship automatically.

. Delete the reciprocal of the synonym (use/used for) relationship automatically.

. Before deleting a hierarchical relationship disconnect the deleting node (the relationship and its reciprocal entry) from the hierarchical tree.

Input Condition: Query.

. Order queries by MAINTR, REL, SECTR.

. Display the relationship name corresponding to the relationship code (REL).

. Permit a direct access to Look up Terms e Relationships form.

. Permit to copy a term from the Look up Terms form and paste it on the form.

Module A44 - Print Thesaurus Terms

Input Condition: Insertion, Update and Deletion.

. Do not permit these operations in this form.

Input Condition: Value of Choice.

. According the value of choice (1, 2, 3 or 4) generate the following reports:

Report A44_1: Order by Class, Term.
Select all terms and their class codes from the table Term and list them ordered by Class and Term. Select from table Classes the corresponding Class Name. Print the class only when the class changes and at the beginning of a new page.

Report A44_2: Order by Term.
Select all terms and their class codes from table Term and list them ordered by Term. Select from table Classes the corresponding Class Name. Print the class only when the class changes and at the beginning of a new page.

Report A44_3: Order by Term Hierarchical Association.
Select all terms from the table Relships by traversing the hierarchical tree in "preorder". Print terms and their corresponding level in the hierarchical tree. Indent each term 3 times its own level.

Report A44_4: Order by Term, Giving all Relationships.
Select all terms from the table Terms and for each term select:
- the corresponding scope notes from the table Notes,
- The corresponding relationship codes and secondary terms from the table Relships.
- The relationship name corresponding to the relationship code from the table RelName.

Appendix D: Selected Triggers from the INP Files

This appendix briefly explains triggers's usage and lists samples of triggers that are actually used in the proposed system forms. The following definitions are mainly based in definitions presented in the Oracle manual "SQL*Forms, Designer's Reference, Version 2.0" (Zussman, 1987).

Trigger Execution and Conventions

- There are two kinds of triggers:
  a. SQL statement, b. SQL*Forms command

- A trigger is composed of one or more trigger steps. A trigger succeeds when all its steps succeeds.

- A trigger step succeeds if it acts on at least one row (SQL statement) or if it executes properly (SQL*Forms command).

- Triggers have three levels or scopes:
  1. Form, 2. Block, 3. Field.

- When a same type of trigger is defined at more than one level, the specific overrides the general.

- The prompt ";Message if value not found :" is used in a INP file to introduce the message displayed on the screen (if any) when the trigger step fails.

- When the message begins with a star means that the normal criteria for success and failure is reversed. Therefore, if the SQL statement returns any row, the step fails.

- Two sentences beginning with the symbol $ in the place of the referred message indicates a success and a failure label respectively. If the actual step succeeds, the step identified by the success label is executed. If the actual step fails, the step identified by the failure label is executed. When a label is blank (there is only the symbol $) the next step to be executed follows a sequential ordering.

- The prompt ";Must value exist Y/N :" states the necessity of a success value to succeed the trigger step. The normal value is (Y)es. When the answer is (N)o, the step does not fail when no row is acted on (a select statement select no row, or a delete statement deletes no row). In this case, the trigger may fail only because of syntax errors.

- A star preceding the "Y" or "N" referred in the previous item means that it will be returned success if the step causes the trigger to

abort. It is meaningful only if the value corresponding to the prompt "Must value exist Y/N" is "Y".

- A form field has the format ":block.name". When the field name is unique in a form, the block name is no needed (ex: ":DOCUMENT.DRN").

- SQL*Form has extended the SQL "select" statement. It added to it the "into" clause that copies the selected value into a form field. After a "into" clause there is no need of a comma to indicate a form field, once only a form field is allowed in this clause.

- A one row and one column dummy table named DUAL is used in select statements to perform form field validation.

SQL*Forms Commands

There are four SQL*Forms commands:

- #EXEMACRO, to execute macro functions and simulate operator strokes
- #COPY, to copy values between form fields and/or global variables
- #ERASE, to release memory space occupied by a global variable
- #HOST, or #OHOST, to execute operating system commands.

Macro Function Codes Used inside a SQL*Form Exemacro Command

Macro function codes are used inside a #EXEMACRO command to perform the desired action. The most used macro function codes in this work are:

- CALL, suspends processing of the current form and displays the form specified. When terminated with and Exit code, it resumes the original form.

- CALLQRY, the same of the previous except that only queries are permitted in the called form.

- ENTQRY, enter query.

- EXEQRY, execute query.

- GOBLK block, moves the cursor to the specified block in the current form.

- GOFLD block.field, moves the cursor to the specified field in the current form.

- NOOP, no operation; does nothing but displays the message "Unrecognized Command".

- NXTBLK, moves the cursor to the first field of the next block in the current form. If the cursor was in the last block, the next block is the first block in the form.

- PRVBLK, previous block.

- EXETRG user_named_trigger, execute a user-named trigger.

- CREREC, create record.

- DELREC, delete record.

- COMMIT, commit.

- NEWFRM form_name, Replaces the current form in memory by the form specified.

Form: A0, Menu Tree

Form Level.

Remark: This trigger redefines the normal exit function on menu tree modules. It calls the Main Menu when the exit function key is pressed, whenever there not exists a more specific Key-Exit trigger.

```
**KEY-EXIT
;SQL>
#EXEMACRO GOBLK A0;

;Message if value not found :

;Must value exist Y/N :
Y
```

Remark: This trigger disables the Oracle menu key (the user must use the menu screen).

```
**KEY-MENU
;SQL>
#EXEMACRO NOOP;

;Message if value not found :

;Must value exist Y/N :
Y
```

Remark: This trigger disables the next block key (the user must use the menu screen).

```
**KEY-NXTBLK
```

```
;SQL>
#EXEMACRO NOOP;

;Message if value not found :

;Must value exist Y/N :
Y
```

Block A0.

```
Remark: This trigger overwrites the form level trigger key-exit.  Exiting
from the Document Control system must be made from the Main Menu.
*KEY-EXIT
;SQL>
#EXEMACRO EXIT;

;Message if value not found :

;Must value exist Y/N :
Y
```

Field Choice.

```
Remark: This trigger branches to a submenu according the selected choice.
**KEY-NXTFLD
/
;SQL>
SELECT 'X' FROM DUAL
WHERE :A0.CHOICE IN ('1','2','3','4')
/
;Message if value not found :
Invalid choice. Type 1, 2, 3, or 4 and Enter.
;Must value exist Y/N :
Y
#EXEMACRO  CASE A0.CHOICE IS
   WHEN '1'            THEN GOBLK A1;
   WHEN '2'            THEN GOBLK A2;
   WHEN '3'            THEN GOBLK A3;
   WHEN '4'            THEN GOBLK A4;
END CASE;

;Message if value not found :

;Must value exist Y/N :
Y
```

Block A2, Field Choice .

```
Remark: This trigger calls other forms according to the selected choice.
**KEY-NXTFLD
/
;SQL>
```

```
SELECT 'X' FROM DUAL
WHERE :A2.CHOICE IN ('1','2','3','4','5','6','7')
/
;Message if value not found :
Invalid choice.
;Must value exist Y/N :
Y
#EXEMACRO  CASE A2.CHOICE IS
   WHEN '1'            THEN CALL A21;
   WHEN '2'            THEN CALL A22;
   WHEN '3'            THEN CALL A23;
   WHEN '4'            THEN CALL A24;
   WHEN '5'            THEN GOBLK A25;
   WHEN '6'            THEN CALL A26;
   WHEN '7'            THEN CALL A27;
END CASE;

;Message if value not found :

;Must value exist Y/N :
Y
```

Form A11, Enter Document

Block Document.

Remark: This trigger executes the following actions:
1) generates the DRN automatically according the operator choice placed in the field Automatic_Number:
"R" - DRN corresponding to a received document
"S" - DRN corresponding to a sent document
Other value - do not generate the DRN
2) Insert a validity date 60 months (5 years) later than the register date.
3) Insert into table Doc_Log and Holder values corresponding to a custody transference from the Protocol Section to the Commander.
4) Insert into the Processing Phase Status the value "P" (processing).
5) Insert into the Archive Section Status the value "U" if the value in field Classif is "U", otherwise insert a "C".

```
*PRE-INSERT
;SQL>
SELECT 'X' FROM DUAL
WHERE :AUTOMATIC_NUMBER = 'R'
/* Failure label TEST_S                    */
/
;Message if value not found :
$REC $TEST_S
;Must value exist Y/N :
N
$REC
```

```
SELECT * FROM SEQNUMBERS
WHERE TABLENAME = 'DOC_RECEIVED'
FOR UPDATE OF LASTNUMBER
/
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE SEQNUMBERS
SET LASTNUMBER = LASTNUMBER + 1
WHERE TABLENAME = 'DOC_RECEIVED'
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT LASTNUMBER
INTO :DOCUMENT.DRN
FROM SEQNUMBERS
WHERE TABLENAME = 'DOC_RECEIVED'
/
;Message if value not found :
$VALIDITY $
;Must value exist Y/N :
Y
$TEST_S
SELECT 'X' FROM dual
WHERE :AUTOMATIC_NUMBER = 'S'
/* Failure label VALIDITY          */
/
;Message if value not found :
$SENT $VALIDITY
;Must value exist Y/N :
N
$SENT
SELECT * FROM SEQNUMBERS
WHERE TABLENAME = 'DOC_SENT'
FOR UPDATE OF LASTNUMBER
/
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE SEQNUMBERS
SET LASTNUMBER = LASTNUMBER + 1
WHERE TABLENAME = 'DOC_SENT'
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT LASTNUMBER
```

```
INTO :DOCUMENT.DRN
FROM SEQNUMBERS
WHERE TABLENAME = 'DOC_SENT'
/
;Message if value not found :
$VALIDITY $
;Must value exist Y/N :
Y
$VALIDITY
SELECT ADD_MONTHS(SYSDATE,60)
INTO :DOCUMENT.VALIDITY
FROM DUAL
/
;Message if value not found :

;Must value exist Y/N :
Y
INSERT INTO DOC_LOG VALUES
(:DOCUMENT.DRN,SYSDATE,TO_CHAR(SYSDATE,'HH24:MI'),'PROTOC','COMMAN')
/
;Message if value not found :

;Must value exist Y/N :
Y
INSERT INTO HOLDER (DRN,DEPCODE) VALUES
 (:DOCUMENT.DRN,'COMMAN')
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT 'X' FROM DUAL
WHERE :DOCUMENT.CLASSIF = 'P'
/* Do not abort trigger when step fails
   Failure label CLASSIFIED */
/
;Message if value not found :
$UNCLASSIFIED $CLASSIFIED
;Must value exist Y/N :
N
$UNCLASSIFIED
SELECT 'U' INTO :DOCUMENT.ARCHSTS
FROM DUAL
/* Success label END */
/
;Message if value not found :
$END $
;Must value exist Y/N :
Y
$CLASSIFIED
SELECT 'C' INTO :DOCUMENT.ARCHSTS
FROM DUAL
```

```
;Message if value not found :

;Must value exist Y/N :
Y



Remark: This trigger forbids deletions in the block
*PRE-DELETE
;SQL>
SELECT 'X' FROM DUAL
/* Abort trigger when step fails
   Reverse return code */

;Message if value not found :
*It is not permitted to delete documents in this module. Use GENERIC
UPDATE.
;Must value exist Y/N :
Y
```

Field Automatic_Number.

```
Remark: This trigger validates the value entered in the field
Automatic_Number against constant values.
**POST-CHANGE
/
;SQL>
SELECT 'X'
FROM DUAL  /* dual is a dummy table */
WHERE :DOCUMENT.AUTOMATIC_NUMBER IN ('R','S','N')
/
;Message if value not found :
Invalid code for DRN automatic generation. Enter R, S, or N.
;Must value exist Y/N :
Y



Remark: This trigger makes the cursor to skip the DRN field if the
operator chose automatic DRN generation (values R or S entered in field
Automatic_Number).
**KEY-NXTFLD
/
;SQL>
#EXEMACRO CASE AUTOMATIC_NUMBER IS
   WHEN 'N'    THEN NXTFLD;
   WHEN OTHERS THEN NXTFLD;NXTFLD;
END CASE;

;Message if value not found :
```

```
;Must value exist Y/N :
Y
```

Field DocType.

```
Remark: This trigger validates the value entered in field DocType against
values selected from the table Doc_Type.
**POST-CHANGE
/
;SQL>
SELECT DOCTYPE
FROM DOC_TYPE
WHERE DOC_TYPE.DOCTYPE = :DOCUMENT.DOCTYPE

;Message if value not found :
Invalid document type. Press <F9> to see a field list.
;Must value exist Y/N :
Y
```

Field OrgName.

```
Remark: This trigger validates the value entered in field OrgName against
values selected from the thesaurus (table Terms).
**POST-CHANGE
/
;SQL>
SELECT TERM
FROM TERMS
WHERE TERMS.TERM = :DOCUMENT.ORGNAME
AND   TERMS.CLASSCD = '01'

;Message if value not found :
Invalid organization name. Enter again.
;Must value exist Y/N :
Y
```

Form A12, Print Processing Sheet, Block A12

```
Remark: This trigger retrieves from table SeqNumbers, before the block is
displayed on the screen, the values of the last received and sent
documents printed and the last received and sent document entered in the
system.
*PRE-BLOCK
;SQL>
SELECT LASTNUMBER
INTO :P_RECEIVED
FROM SEQNUMBERS
WHERE SEQNUMBERS.TABLENAME = 'PRINT_PS_RECEIVED'
/
;Message if value not found :
```

```
;Must value exist Y/N :
Y
SELECT LASTNUMBER
INTO :P_SENT
FROM SEQNUMBERS
WHERE SEQNUMBERS.TABLENAME = 'PRINT_PS_SENT'
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT LASTNUMBER
INTO :E_RECEIVED
FROM SEQNUMBERS
WHERE SEQNUMBERS.TABLENAME = 'DOC_RECEIVED'
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT LASTNUMBER
INTO :E_SENT
FROM SEQNUMBERS
WHERE SEQNUMBERS.TABLENAME = 'DOC_SENT'

;Message if value not found :

;Must value exist Y/N :
Y
```

Field Choice.

Remark: This trigger calls the report generator corresponding to the user choice. After the report generation it is called a script file to update the table SeqNumbers with the last document printed.

```
**KEY-NXTFLD
/
;SQL>
SELECT 'X' FROM DUAL
WHERE :CHOICE = '1'
/
;Message if value not found :
$REC $SENT
;Must value exist Y/N :
N
$REC
#OHOST 'REPORT A12_REC -s'
/
;Message if value not found :

;Must value exist Y/N :
Y
```

```
#OHOST 'SQLPLUS @A12_REC.SQL'
/
;Message if value not found :
$END $
;Must value exist Y/N :
Y
$SENT
SELECT 'X' FROM DUAL
WHERE :CHOICE = '2'
/
;Message if value not found :
$START_REPORT $DRN
;Must value exist Y/N :
N
$START_REPORT
#OHOST 'REPORT A12_SENT -s'
/
;Message if value not found :

;Must value exist Y/N :
Y
#OHOST 'SQLPLUS @A12_SENT.SQL'
/
;Message if value not found :
$END $
;Must value exist Y/N :
Y
$DRN
SELECT 'X' FROM DUAL
WHERE :CHOICE = '3'
/
;Message if value not found :
Invalid choice.  Choose 1, 2, or 3.  For choice 3, type the DRN first.
;Must value exist Y/N :
Y
#OHOST  'REPORT A12_DRN -s'
/
;Message if value not found :

;Must value exist Y/N :
Y
$END
#EXEMACRO NEWFRM A12;

;Message if value not found :

;Must value exist Y/N :
Y
```

Form A21, Generic Update, Block Document

Remark: This trigger makes all the deletions that are necessary for eliminating a document from the database. Observe the tables locking sequence. Tables are unlocked only when the deletion is committed.

```
*PRE-DELETE
;SQL>
LOCK TABLE DOCUMENT IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE REFERENCE IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE ANNEX     IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE DOC_LOG   IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE HOLDER    IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE JOIN      IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE MICROFILM IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE SUBJECT   IN SHARE UPDATE MODE
/
;Message if value not found :
```

```
;Must value exist Y/N :
Y
DELETE FROM TO_DELETE
/
;Message if value not found :

;Must value exist Y/N :
N
INSERT INTO TO_DELETE (DRN)
   SELECT JOINED FROM JOIN
   WHERE MAIN = :DOCUMENT.DRN
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM DOCUMENT
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
N
INSERT INTO TO_DELETE (DRN) VALUES (:DOCUMENT.DRN)
/
;Message if value not found :

;Must value exist Y/N :
Y
DELETE FROM REFERENCE
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM ANNEX
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM DOC_LOG
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM HOLDER
```

```
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM JOIN
WHERE JOINED IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM MICROFILM
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM SUBJECT
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM TO_DELETE

;Message if value not found :

;Must value exist Y/N :
Y
```

Remark: This trigger queries the Reference and Annex blocks automatically, when the Block document is queried.

```
*KEY-ENTQRY
;SQL>
#EXEMACRO ENTQRY;
/
;Message if value not found :

;Must value exist Y/N :
*Y
#EXEMACRO NXTBLK; EXEQRY;
/
;Message if value not found :

;Must value exist Y/N :
N
```

```
#EXEMACRO NXTBLK; EXEQRY;
/
;Message if value not found :

;Must value exist Y/N :
N
#EXEMACRO PRVBLK; PRVBLK;

;Message if value not found :

;Must value exist Y/N :
Y
```

Form A22, Change Validity, Block Document

Remark: This trigger tests the Verifier value entered. If there is an error the update is rolled back.

```
*PRE-UPDATE
;SQL>
SELECT 'X' FROM DUAL
WHERE :VERIFIER = TO_CHAR(mod(power(:DRN,2),97))

;Message if value not found :
Verifier incorrect.
;Must value exist Y/N :
Y
```

Form A23, Process Document Destruction, Block A23

Remark: This trigger makes all integrity tests and operations necessary to process a document that has been destroyed.

```
*PRE-DELETE
;SQL>
SELECT DRN  FROM DOCUMENT
WHERE DRN = :DRN
AND FORMSTS <> 'D'
/
;Message if value not found :
This document has already the Form Status "(D)estroyed".
;Must value exist Y/N :
Y
SELECT 'X' FROM DUAL
WHERE TO_CHAR(mod(power(:DRN,2),97)) = :VERIFIER
/
;Message if value not found :
Verifier incorrect.
;Must value exist Y/N :
Y
$LOCK_TABLES
LOCK TABLE DOCUMENT IN SHARE UPDATE MODE
```

```
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE REFERENCE IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE ANNEX     IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE DOC_LOG   IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE HOLDER    IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE JOIN      IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE MICROFILM IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE SUBJECT   IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT * FROM TO_DELETE
/
;Message if value not found :
$NEXT1 $INSERT_INTO_TO_DELETE
;Must value exist Y/N :
```

```
N
$NEXT1
DELETE FROM TO_DELETE
/
;Message if value not found :

;Must value exist Y/N :
Y
$INSERT_INTO_TO_DELETE
INSERT INTO TO_DELETE (DRN) VALUES (:DRN)
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT * FROM JOIN
WHERE MAIN = :DRN
/
;Message if value not found :
$NEXT2 $DELETE_REFERENCE
;Must value exist Y/N :
N
$NEXT2
INSERT INTO TO_DELETE (DRN)
   SELECT JOINED FROM JOIN
   WHERE MAIN = :DRN
/
;Message if value not found :

;Must value exist Y/N :
Y
$DELETE_REFERENCE
SELECT * FROM REFERENCE
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :
$NEXT3 $DELETE_ANNEX
;Must value exist Y/N :
N
$NEXT3
DELETE FROM REFERENCE
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
Y
$DELETE_ANNEX
SELECT * FROM ANNEX
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :
```

```
$NEXT4 $DELETE_DOC_LOG
;Must value exist Y/N :
N
$NEXT4
DELETE FROM ANNEX
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
Y
$DELETE_DOC_LOG
SELECT * FROM DOC_LOG
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :
$NEXT5 $DELETE_HOLDER
;Must value exist Y/N :
N
$NEXT5
DELETE FROM DOC_LOG
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
Y
$DELETE_HOLDER
SELECT * FROM HOLDER
WHERE DRN IN (SELECT DRN FROM TO_DELETE
              MINUS
              SELECT DRN FROM TO_DELETE
              WHERE DRN = :DRN)
/
;Message if value not found :
$NEXT6 $DELETE_JOIN
;Must value exist Y/N :
N
$NEXT6
DELETE FROM HOLDER
WHERE DRN IN (SELECT DRN FROM TO_DELETE
              MINUS
              SELECT DRN FROM TO_DELETE
              WHERE DRN = :DRN)
/
;Message if value not found :

;Must value exist Y/N :
Y
$DELETE_JOIN
SELECT * FROM JOIN
WHERE JOINED IN (SELECT DRN FROM TO_DELETE)
```

```
/
;Message if value not found :
$NEXT7 $DELETE_MICROFILM
;Must value exist Y/N :
N
$NEXT7
DELETE FROM JOIN
WHERE JOINED IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
Y
$DELETE_MICROFILM
SELECT * FROM MICROFILM
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :
$NEXT8 $DELETE_SUBJECT
;Must value exist Y/N :
N
$NEXT8
DELETE FROM MICROFILM
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
Y
$DELETE_SUBJECT
SELECT * FROM SUBJECT
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :
$NEXT9 $UPDATE_DOCUMENT
;Must value exist Y/N :
N
$NEXT9
DELETE FROM SUBJECT
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :

;Must value exist Y/N :
Y
$UPDATE_DOCUMENT
UPDATE DOCUMENT
SET FORMSTS = 'D'
WHERE DRN IN (SELECT DRN FROM TO_DELETE)
/
;Message if value not found :
```

```
;Must value exist Y/N :
Y
INSERT INTO REFERENCE (DRN, REFER)
  SELECT TO_DELETE.DRN, :REFERENCE
  FROM TO_DELETE
/
;Message if value not found :

;Must value exist Y/N :
Y
DELETE FROM TO_DELETE

;Message if value not found :

;Must value exist Y/N :
Y
```

Form A24, Join Documents, Block Join

```
Remark: This trigger performs the integrity rules necessary to join two
documents.
*PRE-INSERT
;SQL>
SELECT 'X' FROM DUAL
WHERE :MAIN <> :JOINED
/
;Message if value not found :
Main and Joined cannot be the same document.
;Must value exist Y/N :
Y
SELECT * FROM DOCUMENT
WHERE DRN IN (:MAIN,:JOINED)
FOR UPDATE OF JOINSTS
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE DOC_LOG IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE HOLDER IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE DOCUMENT
```

```
SET JOINSTS = 'M'
WHERE DRN = :MAIN
/
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE DOCUMENT
SET JOINSTS = 'J'
WHERE DRN = :JOINED
/
;Message if value not found :

;Must value exist Y/N :
Y
DELETE FROM DOC_LOG
WHERE DRN = :JOINED
/
;Message if value not found :

;Must value exist Y/N :
Y
DELETE FROM HOLDER
WHERE DRN = :JOINED

;Message if value not found :

;Must value exist Y/N :
Y
```

Remark: This trigger executes the integrity rules necessary to separate two documents that were joined before.

```
*PRE-DELETE
;SQL>
SELECT * FROM DOCUMENT
WHERE DRN IN (:MAIN,:JOINED)
FOR UPDATE OF JOINSTS
/
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE DOCUMENT
SET JOINSTS = 'I'
WHERE DRN = :JOINED
/
;Message if value not found :

;Must value exist Y/N :
Y
```

```
INSERT INTO HOLDER (DRN, NAME, DEPCODE)
SELECT :JOINED, NAME, DEPCODE
FROM HOLDER
WHERE DRN = :MAIN
/
;Message if value not found :

;Must value exist Y/N :
Y
INSERT INTO DOC_LOG (DRN, LDATE, LTIME, SEND_DEP, REC_DEP)
SELECT :JOINED, LDATE, LTIME, SEND_DEP, REC_DEP
FROM DOC_LOG_VIEW
WHERE DRN = :MAIN
AND JULIAN_D_T = (SELECT MAX(ALL JULIAN_D_T)
                  FROM DOC_LOG_VIEW
                  WHERE DRN = :MAIN)
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT * FROM JOIN
WHERE MAIN = :MAIN
AND JOINED <> :JOINED
/
;Message if value not found :
$END $
;Must value exist Y/N :
N
UPDATE DOCUMENT
SET JOINSTS = 'I'
WHERE DRN = :MAIN

;Message if value not found :

;Must value exist Y/N :
Y
```

Form A251, Transfer Department Custody, Block Doc Log

Remark: This trigger verifies whether the sending department in a document transference has received the document previously. If it was, set History Status equal "N"ormal, if not, set it equal "A"lteration.

```
*POST-INSERT
;SQL>
SELECT *
FROM DOC_LOG Y
WHERE EXISTS
  (
  SELECT *
  FROM DOC_LOG X
```

```
  WHERE DRN = :DRN
  AND Y.DRN = DRN
  AND Y.SEND_DEP <> REC_DEP
  AND TO_NUMBER
      (TO_CHAR (Y.LDATE,'J') || SUBSTR(Y.LTIME,1,2)
                             || SUBSTR(Y.LTIME,4,2))
      =      (SELECT MIN (ALL TO_NUMBER
                         (TO_CHAR (LDATE,'J')
                         || SUBSTR(LTIME,1,2)
                         || SUBSTR(LTIME,4,2)))
              FROM DOC_LOG
              WHERE DRN = :DRN
              AND TO_CHAR (LDATE,'J') || SUBSTR(LTIME,1,2)
                                      || SUBSTR(LTIME,4,2)
                  >
                  TO_CHAR (X.LDATE,'J') || SUBSTR(X.LTIME,1,2)
                                        || SUBSTR(X.LTIME,4,2)
                  )
         )
/
;Message if value not found :
$SET_ALTERATION $SET_NORMAL
;Must value exist Y/N :
N
$SET_NORMAL
LOCK TABLE DOCUMENT IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE DOCUMENT
SET HISTSTS = 'N'
WHERE DRN = :DOC_LOG.DRN
/
;Message if value not found :
$END $
;Must value exist Y/N :
Y
$SET_ALTERATION
LOCK TABLE DOCUMENT IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE DOCUMENT
SET HISTSTS = 'A'
WHERE DRN = :DOC_LOG.DRN

;Message if value not found :
```

```
;Must value exist Y/N :
Y
```

Field DRN.

```
Remark: This trigger verifies:
      1) Whether the document exists.  If the document already has a row
      in the table Doc_Log, is because it exists, and it is not a joined
      nor a destroyed document.
      2) Whether the document is a joined document in a join
      relationship.
      3) Whether the document is a destroyed document.
**POST-CHANGE\
/
;SQL>
SELECT * FROM DOC_LOG
WHERE DRN = :DRN
/
;Message if value not found :
$END $
;Must value exist Y/N :
N
SELECT * FROM DOCUMENT
WHERE DRN = :DRN
/
;Message if value not found :
This document does not exist.
;Must value exist Y/N :
Y
SELECT * FROM DOCUMENT
WHERE DRN = :DRN
AND JOINSTS = 'J'
/
;Message if value not found :
*This document is Joined.  It cannot be transferred alone.
;Must value exist Y/N :
Y
SELECT * FROM DOCUMENT
WHERE DRN = :DRN
AND FORMSTS = 'D'

;Message if value not found :
*This document was destroyed.
;Must value exist Y/N :
Y
```

Form A322, Look Up Thesaurus Terms and Relationships

Form Level.

Remark: This trigger uses variable reference (&var). It copies the contents of the current field in the form to a global variable. This is used to paste the copied field in another field elsewhere, in the calling module.

```
**KEY-MENU
;SQL>
#COPY &SYSTEM.CURRENT_FIELD GLOBAL.VAR

;Message if value not found :

;Must value exist Y/N :
Y
```

Block Terms, Field ClassCd.

Remark: This trigger selects the class name into the field Class, on the screen, that corresponds to the code entered in the field ClassCd.

```
**POST-CHANGE
/
;SQL>
SELECT CLASS INTO :CLASS
FROM CLASSES
WHERE CLASSCD = :CLASSCD

;Message if value not found :
This class does not exist. Press F9 to see list of values.
;Must value exist Y/N :
Y
```

Form A36, Show Statistics, Block A36

Remark: This trigger accesses several tables to compute statistics. The form does not have a base table.

```
*KEY-MENU
;SQL>
SELECT COUNT(ALL DRN) INTO TOTAL
FROM DOCUMENT
WHERE ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT AVG( ALL ( MAX(ALL LDATE) - MIN(ALL LDATE) + 1) ),
    STDDEV( ALL ( MAX(ALL LDATE) - MIN(ALL LDATE) + 1) )
INTO AVGDAYS, STDDAYS,
```

```
FROM DOC_LOG
WHERE DRN IN (SELECT DRN
              FROM DOCUMENT
              WHERE PROCSTS = 'S'
              AND ISSUEDT
              BETWEEN :FROMDT AND :TODT)
GROUP BY DRN
/* This standard deviation is UNBIASED */
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL FORMSTS) INTO PAPER
FROM DOCUMENT
WHERE FORMSTS = 'P'
AND ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL FORMSTS) INTO MICROFILM
FROM DOCUMENT
WHERE FORMSTS = 'M'
AND ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL FORMSTS) INTO DESTROYED
FROM DOCUMENT
WHERE FORMSTS = 'D'
AND ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL JOINSTS) INTO INDEPENDENT
FROM DOCUMENT
WHERE JOINSTS = 'I'
AND ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL JOINSTS) INTO MAIN
FROM DOCUMENT
WHERE JOINSTS = 'M'
```

```
AND ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL JOINSTS) INTO JOINED
FROM DOCUMENT
WHERE JOINSTS = 'J'
AND ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL PROCSTS) INTO PROCESSING
FROM DOCUMENT
WHERE PROCSTS = 'P'
AND ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL PROCSTS) INTO WAITING
FROM DOCUMENT
WHERE PROCSTS = 'W'
AND ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL PROCSTS) INTO SOLVED
FROM DOCUMENT
WHERE PROCSTS = 'S'
AND ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL HISTSTS) INTO NORMAL
FROM DOCUMENT
WHERE HISTSTS = 'N'
AND ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL HISTSTS) INTO ALTERATION
```

```
FROM DOCUMENT
WHERE HISTSTS = 'A'
AND ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL ARCHSTS) INTO UNCLASSIFIED
FROM DOCUMENT
WHERE ARCHSTS = 'U'
AND ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL ARCHSTS) INTO CLASSIFIED
FROM DOCUMENT
WHERE ARCHSTS = 'C'
AND ISSUEDT BETWEEN :FROMDT AND :TODT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT
(:PAPER/:TOTAL)*100, (:MICROFILM/:TOTAL)*100, (:DESTROYED/:TOTAL)*100,
(:INDEPENDENT/:TOTAL)*100, (:MAIN/:TOTAL)*100, (:JOINED/:TOTAL)*100,
(:PROCESSING/:TOTAL)*100, (:WAITING/:TOTAL)*100, (:SOLVED/:TOTAL)*100,
(:NORMAL/:TOTAL)*100, (:ALTERATION/:TOTAL)*100,
(:CLASSIFIED/:TOTAL)*100, (:UNCLASSIFIED/:TOTAL)*100
INTO
PPAPER, PMICROFILM, PDESTROYED,
PINDEPENDENT, PMAIN, PJOINED,
PPROCESSING, PWAITING, PSOLVED,
PNORMAL, PALTERATION,
PCLASSIFIED, PUNCLASSIFIED
FROM DUAL
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL CLASSCD) INTO CLASSES
FROM CLASSES
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL TERM) INTO TERMS
```

```
FROM TERMS
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL NOTE) INTO NOTES
FROM NOTES
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL MAINTR)
INTO RELSHIPS
FROM RELSHIPS
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(DISTINCT DRN) INTO INDEXED
FROM SUBJECT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT (:INDEXED/:TOTAL) * 100
INTO PINDEXED
FROM DUAL
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL DRN) / COUNT(DISTINCT DRN) AVERAGE
INTO AVGTERM
FROM SUBJECT
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT COUNT(ALL DRN) / COUNT(DISTINCT DRN) AVERAGE,
       STDDEV ( ALL (COUNT(ALL DRN) / COUNT(DISTINCT DRN)) )
                                                    "STANDARD DEV"
INTO STDTERM
FROM SUBJECT
GROUP BY DRN

;Message if value not found :
```

```
;Must value exist Y/N :
Y
```

Form A41, Maintain Classes, Block Classes

```
Remark: This trigger does the following automatic insertions:
      1) Insert the class name as a term in the table TERMS.
      2) Insert the scope note "class".
      3) Connect the term to the root of the hierarchical tree.
*PRE-INSERT
;SQL>
INSERT INTO TERMS (TERM,CLASSCD)
VALUES (:CLASS,:CLASSCD)
/* Insert the class name as a term in the table TERMS
/
;Message if value not found :

;Must value exist Y/N :
Y
INSERT INTO NOTES (TERM,NOTE)
VALUES (:CLASS,'CLASS')
/* Insert the word CLASS as a scope note for the term that
/* corresponds to a class
/
;Message if value not found :

;Must value exist Y/N :
Y
INSERT INTO RELSHIPS (MAINTR,REL,SECTR)
VALUES (:CLASS,'BT','ROOT')
/* Create a hierarchical relationship having ROOT as the parent and the
/* class being inserted as the child
/
;Message if value not found :

;Must value exist Y/N :
Y
INSERT INTO RELSHIPS (MAINTR,REL,SECTR)
VALUES ('ROOT','NT',:CLASS)
/* Create a hierarchical relationship having ROOT as the parent and the
/* class being inserted as the child

;Message if value not found :

;Must value exist Y/N :
Y
```

```
Remark: This trigger verifies whether the user is trying to update a
system maintained class and performs integrity operations related to the
update being made in the table class.
*PRE-UPDATE
;SQL>
SELECT * FROM CLASSES
WHERE ROWID = :ROWID
AND  (CLASSCD = '00' OR CLASSCD = '01')
/* Do not permit to update classes 00 or 01
/
;Message if value not found :
*This class is owned by the system. Update is not allowed.
;Must value exist Y/N :
Y
LOCK TABLE SUBJECT   IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE SUBJECT
SET TERM = :CLASS
WHERE TERM = (SELECT CLASS FROM CLASSES
              WHERE ROWID = :ROWID)
/* update all rows that have the old class name as the term
/
;Message if value not found :

;Must value exist Y/N :
N
LOCK TABLE TERMS IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE TERMS SET TERM = :CLASS
WHERE TERM = (SELECT CLASS FROM CLASSES
              WHERE ROWID = :ROWID)
/* Update the corresponding term in the table TERMS
/
;Message if value not found :

;Must value exist Y/N :
N
SELECT 'X' FROM DUAL
WHERE
:CLASSCD = (SELECT CLASSCD FROM CLASSES
            WHERE ROWID = :ROWID)
/* it is being tested whether the class code was changed. If not
/* there is no necessity of updating the class code of terms
/* that pertain to the class being updated
```

```
/* if the step succeeds, branch to update table Notes
/
;Message if value not found :
$UPDATE_NOTES $
;Must value exist Y/N :
N
UPDATE TERMS
SET CLASSCD = :CLASSCD
WHERE CLASSCD = (SELECT CLASSCD FROM CLASSES
                 WHERE ROWID = :ROWID)
/* Update all rows in table Terms that have the same class code
/
;Message if value not found :

;Must value exist Y/N :
N
$UPDATE_NOTES
LOCK TABLE NOTES IN SHARE UPDATE MODE
/* Update the term in the table Relships
/
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE NOTES
SET TERM = :CLASS
WHERE TERM = (SELECT CLASS FROM CLASSES
              WHERE ROWID = :ROWID)
/* As a class is also a term, it may have scope notes to be updated
/
;Message if value not found :

;Must value exist Y/N :
N
LOCK TABLE RELSHIPS   IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE RELSHIPS
SET MAINTR = :CLASS
WHERE MAINTR = (SELECT CLASS FROM CLASSES
                WHERE ROWID = :ROWID)
/* update all rows that have the old class name as the main term
/
;Message if value not found :

;Must value exist Y/N :
N
UPDATE RELSHIPS
SET SECTR = :CLASS
```

```
WHERE SECTR = (SELECT CLASS FROM CLASSES
               WHERE ROWID = :ROWID)
/* update all rows that have the old class name as the secondary term

;Message if value not found :

;Must value exist Y/N :
N
```

Remark: This trigger verifies whether the user is trying to delete a system maintained class and performs integrity operations related to the deletion being made in the table class.

```
*PRE-DELETE
;SQL>
SELECT * FROM CLASSES
WHERE ROWID = :ROWID
AND  (CLASSCD = '00' OR CLASSCD = '01')
/* Do not permit to delete classes 00 or 01
/
;Message if value not found :
*This class is owned by the system. Deleting is not allowed.
;Must value exist Y/N :
Y
LOCK TABLE SUBJECT IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
DELETE FROM SUBJECT
WHERE TERM IN (SELECT TERM FROM TERMS
               WHERE CLASSCD = :CLASSCD)
/
;Message if value not found :

;Must value exist Y/N :
N
LOCK TABLE TERMS IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE NOTES IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
DELETE FROM NOTES
```

```
WHERE TERM IN (SELECT TERM FROM TERMS
               WHERE CLASSCD = :CLASSCD)
/
;Message if value not found :

;Must value exist Y/N :
N
LOCK TABLE RELSHIPS IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
DELETE FROM RELSHIPS
WHERE MAINTR IN (SELECT TERM FROM TERMS
                 WHERE CLASSCD = :CLASSCD)
OR    SECTR  IN (SELECT TERM FROM TERMS
                 WHERE CLASSCD = :CLASSCD)
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM TERMS
WHERE CLASSCD = :CLASSCD

;Message if value not found :

;Must value exist Y/N :
N
```

Remark: This trigger is an example of redefining a key to commit an operation immediately after executing it. This avoids the accumulation of operations not committed that would be all rolled back in an eventual error. This also releases the locks requested by triggers. This redefinition is also used in the "key-delrec" trigger.

```
*KEY-CREREC
;SQL>
#EXEMACRO CREREC; COMMIT;

;Message if value not found :

;Must value exist Y/N :
Y
```

Form A42, Maintain Terms, Block Terms

```
Remark: This trigger performs integrity operations related with a term
update.
*PRE-UPDATE
;SQL>
SELECT * FROM CLASSES
WHERE CLASS = ( SELECT TERM FROM TERMS
                WHERE ROWID = :TERMS.ROWID)
/* do not permit to update a term that corresponds to a class
/
;Message if value not found :
*Term corresponds to a class name. Use A41-Maintain Classes.
;Must value exist Y/N :
Y
LOCK TABLE SUBJECT IN SHARE UPDATE MODE
/* Update the term in the table Subject.
/* As the old name is not anymore in the form, we use the rowid to
/* find the matches in the table Subject.
/
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE SUBJECT
SET TERM = :TERMS.TERM
WHERE TERM = (SELECT TERM FROM TERMS
              WHERE TERMS.ROWID = :TERMS.ROWID)
/
;Message if value not found :

;Must value exist Y/N :
N
LOCK TABLE NOTES IN SHARE UPDATE MODE
/* Update the term in the table Relships
/
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE NOTES
SET TERM = :TERMS.TERM
WHERE TERM = (SELECT TERM FROM TERMS
              WHERE TERMS.ROWID = :TERMS.ROWID)
/
;Message if value not found :

;Must value exist Y/N :
N
LOCK TABLE RELSHIPS IN SHARE UPDATE MODE
/* Update the term in the table Relships
/
```

```
;Message if value not found :

;Must value exist Y/N :
Y
UPDATE RELSHIPS
SET MAINTR = :TERMS.TERM
WHERE MAINTR = (SELECT TERM FROM TERMS
                WHERE TERMS.ROWID = :TERMS.ROWID)
/
;Message if value not found :

;Must value exist Y/N :
N
UPDATE RELSHIPS
SET SECTR = :TERMS.TERM
WHERE SECTR = (SELECT TERM FROM TERMS
                WHERE TERMS.ROWID = :TERMS.ROWID)

;Message if value not found :

;Must value exist Y/N :
N



Remark: This trigger performs integrity operations related with a term
deletion.
*PRE-DELETE
;SQL>
SELECT * FROM CLASSES
WHERE CLASS = ( SELECT TERM FROM TERMS
                WHERE ROWID = :TERMS.ROWID)
/* do not permit to delete the term that corresponds to a class
/
;Message if value not found :
*Term corresponds to a class name. Use A41-Maintain Classes.
;Must value exist Y/N :
Y
LOCK TABLE SUBJECT IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
DELETE FROM SUBJECT
WHERE TERM = :TERMS.TERM
/
;Message if value not found :

;Must value exist Y/N :
N
LOCK TABLE NOTES IN SHARE UPDATE MODE
```

```
/
;Message if value not found :

;Must value exist Y/N :
Y
LOCK TABLE RELSHIPS IN SHARE UPDATE MODE
/
;Message if value not found :

;Must value exist Y/N :
Y
$VERIFY_RELSHIPS
SELECT MAINTR, REL, SECTR
INTO MAINTR, REL, SECTR
FROM RELSHIPS
WHERE MAINTR = :TERMS.TERM
AND REL = 'BT'
/* verify if the term being deleted is involved in a hierarchical
/* relationship. If it is, the values of maintr, rel and sectr are
/* copied to control fields for the purpose of disconnecting the
/* term from the hierarchical tree and the disconnecting step is called
/* if the term does not have relationships, proceed to delete
/* all rows that have either the main or secondary term equal
/* the term being deleted
/
;Message if value not found :
$DISCONNECT $DEL_RELSHIPS
;Must value exist Y/N :
N
; call trigger to disconnect the term from a node in the
; table Relships. Only one node is disconnected at a time.
; When returning, deviate to the beginning to verify
; whether still exists any hierarchical relationships
; Keep looping until all hierar. rel. were disconnected.
$DISCONNECT
#EXEMACRO EXETRG DISCONNECT_NODE;
/
;Message if value not found :
$VERIFY_RELSHIPS $
;Must value exist Y/N :
Y
$DEL_RELSHIPS
DELETE FROM RELSHIPS
WHERE MAINTR = :TERMS.TERM
OR SECTR = :TERMS.TERM
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM NOTES
WHERE TERM = :TERMS.TERM
```

```
/* Do not abort trigger when step fails */

;Message if value not found :

;Must value exist Y/N :
N
```

Remark: This trigger uses variable reference (&var). It copies the contents of a global variable to the current field in the form. This is used to paste a field copied from this or from another form inside a field in this module. See the following trigger and also a trigger in form A322 that are used in combination with this one.

```
*KEY-CREREC
;SQL>
#COPY GLOBAL.VAR :TERMS.TERM

;Message if value not found :

;Must value exist Y/N :
Y
```

Remark: This trigger is used in combination with the previous trigger to "cut and paste" field values.

```
;SQL>
#COPY :TERMS.TERM GLOBAL.VAR

;Message if value not found :

;Must value exist Y/N :
Y
```

Remark: This trigger is used to disconnect a hierarchical relationship in the table Relships. This trigger is called from another trigger.

```
*DISCONNECT_NODE
;SQL>
; disconnect a hierarchical relationship from the
; hierarchical tree
UPDATE RELSHIPS
SET MAINTR = :SECTR
WHERE MAINTR = :MAINTR
AND REL = 'NT'
/* If the term has a child, connect its child to its parent
/
;Message if value not found :
```

```
;Must value exist Y/N :
N
UPDATE RELSHIPS
SET SECTR = :SECTR
WHERE SECTR = :MAINTR
AND REL = 'BT'
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM RELSHIPS
WHERE MAINTR = :MAINTR AND SECTR = :SECTR AND REL = 'BT'
/* delete the relationship
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM RELSHIPS
WHERE MAINTR = :SECTR AND SECTR = :MAINTR AND REL = 'NT'
/* If there exists, deletes the reciprocal

;Message if value not found :

;Must value exist Y/N :
N
```

Form A43, Maintain Relationships, Block Relships

Remark: This form uses user-defined triggers in combination with pre-insert, pre-update and pre-delete triggers to enforce integrity rules in insertions, updates and deletions executed in the table Relships. Each kind of relationship has its particular rules. A case statement identifies the relation involved and calls the appropriated trigger.

```
*PRE-INSERT
;SQL>
#EXEMACRO CASE :REL IS
   WHEN 'RT'  THEN EXETRG INSERT_RT;
   WHEN 'US'  THEN EXETRG INSERT_US;
   WHEN 'UF'  THEN EXETRG INSERT_UF;
   WHEN 'BT'  THEN EXETRG INSERT_BT;
   WHEN 'NT'  THEN EXETRG INSERT_NT;
END CASE;

;Message if value not found :

;Must value exist Y/N :
Y
```

```
*PRE-UPDATE
;SQL>
SELECT * FROM RELSHIPS
WHERE ROWID = :ROWID
AND (MAINTR = 'ROOT' OR SECTR = 'ROOT')
/
;Message if value not found :
*Relationship owned by the system. Updating not allowed.
;Must value exist Y/N :
Y
#EXEMACRO CASE :REL IS
   WHEN 'RT'  THEN EXETRG UPDATE_RT;
   WHEN 'US'  THEN EXETRG UPDATE_US;
   WHEN 'UF'  THEN EXETRG UPDATE_UF;
   WHEN 'BT'  THEN EXETRG UPDATE_BT;
   WHEN 'NT'  THEN EXETRG UPDATE_NT;
END CASE;

;Message if value not found :

;Must value exist Y/N :
Y



*PRE-DELETE
;SQL>
#EXEMACRO CASE :REL IS
   WHEN 'RT'  THEN EXETRG DELRT;
   WHEN 'US'  THEN EXETRG DELUS;
   WHEN 'UF'  THEN EXETRG DELUF;
   WHEN 'BT'  THEN EXETRG DELBT;
   WHEN 'NT'  THEN EXETRG DELNT;
END CASE;

;Message if value not found :

;Must value exist Y/N :
Y



*INSERT_BT
;SQL>
INSERT INTO RELSHIPS VALUES
  (:SECTR, 'NT', :MAINTR)
/* Insert the reciprocal of the relationship
/
;Message if value not found :
```

```
;Must value exist Y/N :
Y
SELECT 'X' FROM RELSHIPS
WHERE (:SECTR = 'ROOT') OR (MAINTR = :SECTR AND REL = 'BT')
/* If the secondary term already has a BROADER TERM
/* relationship goto END
/* Otherwise, we connect the sectr to the class that it pertains
/* This action is intended to not permit hierarchical relationships
/* that are disconnected from the hierarchical tree
/
;Message if value not found :
$END $
;Must value exist Y/N :
N
INSERT INTO RELSHIPS (MAINTR,REL,SECTR)
SELECT :SECTR, 'BT', CLASS
FROM CLASSES
WHERE CLASSCD = :SECCLASSCD
/* Insert a BROADER TERM relationship that has the secondary term
/* being inserted as the main term and its class as the
/* secondary term
/
;Message if value not found :

;Must value exist Y/N :
Y
INSERT INTO RELSHIPS (MAINTR,REL,SECTR)
SELECT CLASS, 'NT', :SECTR
FROM CLASSES
WHERE CLASSCD = :SECCLASSCD
/* Insert the reciprocal

;Message if value not found :

;Must value exist Y/N :
Y



*INSERT_NT
;SQL>
INSERT INTO RELSHIPS VALUES
  (:SECTR, 'BT', :MAINTR)
/* Insert the reciprocal of the relationship
/
;Message if value not found :

;Must value exist Y/N :
Y
SELECT 'X' FROM RELSHIPS
WHERE (:MAINTR = 'ROOT') OR (MAINTR = :MAINTR AND REL = 'BT')
/* If the main term has a BROADER TERM relationship goto END
```

```
/
;Message if value not found :
$END $
;Must value exist Y/N :
N
INSERT INTO RELSHIPS (MAINTR,REL,SECTR)
SELECT :MAINTR, 'BT', CLASS
FROM CLASSES
WHERE CLASSCD = :MAINCLASSCD
/* Insert a BROADER TERM relationship that has the class of the term
/* being inserted as its BROADER TERM
/
;Message if value not found :

;Must value exist Y/N :
Y
INSERT INTO RELSHIPS (MAINTR,REL,SECTR)
SELECT CLASS, 'NT', :MAINTR
FROM CLASSES
WHERE CLASSCD = :MAINCLASSCD
/* Insert the reciprocal

;Message if value not found :

;Must value exist Y/N :
Y



*INSERT_RT
;SQL>
INSERT INTO RELSHIPS VALUES
  (:SECTR, 'RT', :MAINTR)
/* Insert the reciprocal associative entry

;Message if value not found :

;Must value exist Y/N :
Y



*INSERT_UF
;SQL>
INSERT INTO RELSHIPS VALUES
  (:SECTR, 'US', :MAINTR)
/* Insert the reciprocal and goto END

;Message if value not found :

;Must value exist Y/N :
Y
```

```
*INSERT_US
;SQL>
INSERT INTO RELSHIPS VALUES
  (:SECTR, 'UF', :MAINTR)
/* Insert the reciprocal

;Message if value not found :

;Must value exist Y/N :
Y



*UPDATE_BT
;SQL>
SELECT 'X' FROM DUAL
WHERE :MAINCLASSCD = :SECCLASSCD
/* If both terms in a hierarchical relat. do not pertain to the same
/* class give error message
/
;Message if value not found :
In a hierarchical relationship both terms must pertain to the same class.
;Must value exist Y/N :
Y
UPDATE RELSHIPS
SET SECTR = :MAINTR
WHERE MAINTR = :SECTR
AND REL = 'NT'
AND SECTR = (SELECT MAINTR FROM RELSHIPS
             WHERE ROWID = :ROWID)
/* Update the reciprocal of the relationship

;Message if value not found :

;Must value exist Y/N :
Y



*UPDATE_NT
;SQL>
SELECT 'X' FROM DUAL
WHERE :MAINCLASSCD = :SECCLASSCD
/* If both terms in a hierarchical relat. do not pertain to the same
/* class give error message
/
;Message if value not found :
In a hierarchical relationship both terms must pertain to the same class.
;Must value exist Y/N :
```

```
Y
UPDATE RELSHIPS
SET SECTR = :MAINTR
WHERE MAINTR = :SECTR
AND REL = 'BT'
AND SECTR = (SELECT MAINTR FROM RELSHIPS
             WHERE ROWID = :ROWID)
/* Update the reciprocal of the relationship

;Message if value not found :

;Must value exist Y/N :
Y



*UPDATE_RT
;SQL>
UPDATE RELSHIPS
SET SECTR = :MAINTR
WHERE MAINTR = :SECTR
AND REL = 'RT'
AND SECTR = (SELECT MAINTR FROM RELSHIPS
             WHERE ROWID = :ROWID)
/* Update the reciprocal associative entry

;Message if value not found :

;Must value exist Y/N :
Y



*UPDATE_UF
;SQL>
UPDATE RELSHIPS
SET SECTR = :MAINTR
WHERE MAINTR = :SECTR
AND REL = 'US'
AND SECTR = (SELECT MAINTR FROM RELSHIPS
             WHERE ROWID = :ROWID)
/* Update the reciprocal

;Message if value not found :

;Must value exist Y/N :
Y



*UPDATE_US
;SQL>
```

```
UPDATE RELSHIPS
SET SECTR = :MAINTR
WHERE MAINTR = :SECTR
AND REL = 'UF'
AND SECTR = (SELECT MAINTR FROM RELSHIPS
             WHERE ROWID = :ROWID)
/* Update the reciprocal

;Message if value not found :

;Must value exist Y/N :
Y



*DELBT
;SQL>
; disconnect a hierarchical relationship being deleted from the
; hierarchical tree
UPDATE RELSHIPS
SET MAINTR = :SECTR
WHERE MAINTR = :MAINTR
AND REL = 'NT'
/
;Message if value not found :

;Must value exist Y/N :
N
UPDATE RELSHIPS
SET SECTR = :SECTR
WHERE SECTR = :MAINTR
AND REL = 'BT'
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM RELSHIPS
WHERE MAINTR = :SECTR AND SECTR = :MAINTR AND REL = 'NT'
/* If there exists, deletes the reciprocal

;Message if value not found :

;M  t value exist Y/N :
N



*DELNT
;SQL>
; disconnect a hierarchical relationship being deleted from the
; hierarchical tree
```

```
UPDATE RELSHIPS
SET MAINTR = :SECTR
WHERE MAINTR = :MAINTR
AND REL = 'BT'
/
;Message if value not found :

;Must value exist Y/N :
N
UPDATE RELSHIPS
SET SECTR = :MAINTR
WHERE SECTR = :MAINTR
AND REL = 'NT'
/
;Message if value not found :

;Must value exist Y/N :
N
DELETE FROM RELSHIPS
WHERE MAINTR = :SECTR AND SECTR = :MAINTR AND REL = 'BT'
/* If there exists, deletes the reciprocal

;Message if value not found :

;Must value exist Y/N :
N



*DELRT
;SQL>
DELETE FROM RELSHIPS
WHERE MAINTR = :SECTR AND SECTR = :MAINTR AND REL = 'RT'
/* If there exists, deletes the reciprocal

;Message if value not found :

;Must value exist Y/N :
N



*DELUF
;SQL>
DELETE FROM RELSHIPS
WHERE MAINTR = :SECTR AND SECTR = :MAINTR AND REL = 'US'
/* If there exists, deletes the reciprocal
/* go to END

;Message if value not found :
```

```
;Must value exist Y/N :
N



*DELUS
;SQL>
DELETE FROM RELSHIPS
WHERE MAINTR = :SECTR AND SECTR = :MAINTR AND REL = 'UF'
/* If there exists, deletes the reciprocal

;Message if value not found :

;Must value exist Y/N :
N



Remark: The user-triggers with prefix "POST" are called from a post-
change trigger defined for the field SecTr in this same form.  Thus,
these user-triggers are defined at the form level while the calling
trigger is defined at the field level.
*POST_BT
;SQL>
SELECT 'X' FROM DUAL
WHERE :MAINCLASSCD = :SECCLASSCD
/* If both terms in a hierarchical relat. do not pertain to the same
/* class give error message
/
;Message if value not found :
In a hierarchical relationship both terms must pertain to the same class.
;Must value exist Y/N :
Y
SELECT 'X' FROM RELSHIPS
WHERE MAINTR = :MAINTR AND REL = 'BT'
/* A term cannot have more than one BROADER TERM relationship
/
;Message if value not found :
*The main term already has a BROADER TERM relationship. It cannot have
more.
;Must value exist Y/N :
Y
SELECT 'X' FROM RELSHIPS
WHERE REL = 'US' AND (MAINTR = :MAINTR OR MAINTR = :SECTR)
/* If either term already has a USE relationship give a not allowed
message

;Message if value not found :
*The main term or the secondary term has a USE relationship.
;Must value exist Y/N :
Y
```

```
*POST_NT
;SQL>
SELECT 'X' FROM DUAL
WHERE :MAINCLASSCD = :SECCLASSCD
/* If both terms in a hierarchical relat. do not pertain to the same
/* class give error message
/
;Message if value not found :
In a hierarchical relationship both terms must pertain to the same class.
;Must value exist Y/N :
Y
SELECT * FROM RELSHIPS
WHERE SECTR = :SECTR AND REL = 'NT'
/* A secondary term cannot have more than one NARROWER TERM relationship
/
;Message if value not found :
*The secondary term already has a NARROWER TERM relat. It cannot have
more.
;Must value exist Y/N :
Y
SELECT 'X' FROM RELSHIPS
WHERE REL = 'US' AND (MAINTR = :MAINTR OR MAINTR = :SECTR)
/* If either term already has a USE relationship give a not allowed
message

;Message if value not found :
*The main term or the secondary term has a USE relationship.
;Must value exist Y/N :
Y



*POST_UF
;SQL>
SELECT * FROM RELSHIPS
WHERE MAINTR = :MAINTR AND REL = 'US'
/* The main term cannot have a USED FOR relationship if it already has
/* a USE relationship

;Message if value not found :
*The main term already has a USE relationship. It cannot have a USED FOR
rel.
;Must value exist Y/N :
Y



*POST_US
;SQL>
SELECT * FROM RELSHIPS
```

```
WHERE REL = 'US' AND MAINTR = :MAINTR
/* A main term cannot have more than one USE relationship

;Message if value not found :
*The main term already has a "USE" relationship.
;Must value exist Y/N :
Y
```

Field SecTr.

```
Remark: This trigger executes some integrity operations related with the
entered value of secondary term.  It calls user-triggers that were
defined at the form level.
**POST-CHANGE
/
;SQL>
SELECT CLASSCD INTO SECCLASSCD
FROM TERMS
WHERE TERM = :SECTR
/
;Message if value not found :
The secondary term does not exist.
;Must value exist Y/N :
Y
#EXEMACRO CASE :REL IS
   WHEN 'US'  THEN EXETRG POST_US;
   WHEN 'UF'  THEN EXETRG POST_UF;
   WHEN 'BT'  THEN EXETRG POST_BT;
   WHEN 'NT'  THEN EXETRG POST_NT;
END CASE;

;Message if value not found :

;Must value exist Y/N :
Y
```

Form A44, Print Thesaurus Terms, Block A44

```
Remark: This user-triggers are called by a case statement placed in a
next field key trigger (key-nxtfld) pertaining to field Choice.  They
exemplify the use of host commands to execute a report generator program
from inside a form.
*CHOICE1
;SQL>
#OHOST 'REPORT A44_1 -s'
/
;Message if value not found :

;Must value exist Y/N :
Y
#EXEMACRO NEWFRM A44;
```

```
;Message if value not found :

;Must value exist Y/N :
Y



*CHOICE2
;SQL>
#OHOST 'REPORT A44_2 -s'
/
;Message if value not found :

;Must value exist Y/N :
Y
#EXEMACRO NEWFRM A44;

;Message if value not found :

;Must value exist Y/N :
Y



*CHOICE3
;SQL>
#OHOST 'REPORT A44_3 -s'
/
;Message if value not found :

;Must value exist Y/N :
Y
#EXEMACRO NEWFRM A44;

;Message if value not found :

;Must value exist Y/N :
Y



*CHOICE4
;SQL>
#OHOST 'REPORT A44_4 -s'
/
;Message if value not found :

;Must value exist Y/N :
Y
#EXEMACRO NEWFRM A44;
```

```
;Message if value not found :

;Must value exist Y/N :
Y
```

Field Choice.

```
Remark: This trigger validates the user choice and calls the user-trigger
corresponding to the choice, to generate the desired printed report.
**KEY-NXTFLD
/
;SQL>
SELECT 'X' FROM DUAL
WHERE :CHOICE IN ('1','2',':','4')
/
;Message if value not found :
Invalid choice. Type 1, 2, 3, or 4 and Enter.
;Must value exist Y/N :
Y
#EXEMACRO  CASE CHOICE IS
   WHEN '1'            THEN EXETRG CHOICE1;
   WHEN '2'            THEN EXETRG CHOICE2;
   WHEN '3'            THEN EXETRG CHOICE3;
   WHEN '4'            THEN EXETRG CHOICE4;
   WHEN OTHERS         THEN NULL;
END CASE;

;Message if value not found :

;Must value exist Y/N :
Y
```

Appendix E: Report Generation Files

File Report.Bat

SQL*Report is a Oracle's tool composed of two programs used together to retrieve information from the database and to format the derived information as desired. They are the Report Generator (RPT) and the Report Text Formatter (RPT). The following batch file was used to coordinate the two programs and generate the desired reports from inside the forms.

```
ECHO OFF
REM  File Name: REPORT.BAT    Author: Lt Col Silva   Date: Feb 1989
REM  This file is used by:
REM  FORM A12 TO PROCESS THE A12_1.RPT, A12_2.RPT,AND A12_3.RPT
REM  REPORT FILES, WHICH GENERATE PROCESSING SHEETS
REM
REM  FORM A44 TO PROCESS THE A44_1.RPT, A44_2.RPT, A44_3.RPT, AND
REM  A44_4.RPT REPORT FILES, WHICH GENERATE THESAURUS TERM LISTS.
REM
REM  IT CAN ALSO BE USED TO PROCESS ANY OTHER .RPT FILES.
REM  THE REPORT GENERATOR PRODUCES A INTERMEDIATE FILE THAT HAS .RPF
REM  EXTENSION AND THE FINAL REPORT, THAT HAS A .LIS EXTENSION
CLEAR
ECHO.
ECHO  PROCESSING SHEET GENERATION
ECHO.
ECHO  Author Lt Col Silva    Feb 1989
ECHO  Working ...
if (%1)==() goto ERROR
RPT %1.RPT %1.RPF SYSTEM/MANAGER
RPF %1.RPF %1.LIS %2
ECHO.
ECHO Output file is %1.LIS
GOTO END
:ERROR
ECHO  Syntax is  "REPORT  filename"  [-S]
ECHO.
ECHO  filename is a .RPT file, without the extension
ECHO  the report output is directed to filename.lis
ECHO  -s   if used, the output is also directed to the line printer
:END
```

File A12_1.RPT, Received Document Process Sheet Generation

```
.REM                PROCESS SHEET GENERATION  - RECEIVED DOCUMENTS
.REM PRINTS FROM
.REM        THE LAST DOCUMENT THAT HAD ITS PROCESSING SHEET PRINTED
.REM TO
.REM        THE LAST DOCUMENT ENTERED IN THE SYSTEM
.REM

.REM        DEFINE TAB STOPS (TABLES) ON THE REPORT
#DT 1   08 73 #
#DT 2   08 17  18 18  19 20  21 21  22 26  27 27  28 29  55 73 #
#DT 3   08 44  45 45  46 59  60 72 #
#dt 4   08 26  28 33  35 73 #
#DT 5   12 19  20 20  26 32  33 33  54 73 #
#DT 6   20 20  33 33 #
#PAGE 0 60
.REM
.REM                    DECLARE VARIABLES
.REM
.DECLARE TODAY         A9
.DECLARE CLASSIF_NAME A12
.DECLARE CLASSIF       A1
.DECLARE CDRN          A8
.DECLARE DRN           99999999
.DECLARE P_SC          A2
.DECLARE P_OC          A2
.DECLARE P_NUMBER      A5
.DECLARE P_YEAR        A2
.DECLARE IDNR          A5
.DECLARE IDCOMPL       A13
.DECLARE ORGNAME       A20
.DECLARE ISSUEDT       A9
.DECLARE TITLE         A78
.DECLARE DOCTYPE       A7
.DECLARE VALIDITY      A9
.DECLARE VERIFIER      A2

.DECLARE FIRST_DOC_TO_PRINT 99999999
.DECLARE LAST_DOC_TO_PRINT  99999999

.REM                                SELECT MACROS
.REM
.DEFINE           FIRST

      SELECT LASTNUMBER
      INTO   FIRST_DOC_TO_PRINT
      FROM SEQNUMBERS
      WHERE TABLENAME = 'PRINT_PS_RECEIVED'
..
```

```
.DEFINE          LAST

      SELECT LASTNUMBER
      INTO   LAST_DOC_TO_PRINT
      FROM SEQNUMBERS
      WHERE TABLENAME = 'DOC_RECEIVED'
..

.DEFINE          DOCUMENT
.REM
        SELECT  SYSDATE,
                CLASSIF,
                TO_CHAR(DRN),
                TO_CHAR(mod(power(DRN,2),97)),
                DRN,
                TO_CHAR(P_SC),
                TO_CHAR(P_OC),
                TO_CHAR(P_NUMBER),
                TO_CHAR(P_YEAR),
                TO_CHAR(IDNR),
                IDCOMPL,
                ORGNAME,
                ISSUEDT,
                TITLE,
                DOCTYPE,
                VALIDITY
      INTO      TODAY,
                CLASSIF,
                CDRN,
                VERIFIER,
                DRN,
                P_SC,
                P_OC,
                P_NUMBER,
                P_YEAR,
                IDNR,
                IDCOMPL,
                ORGNAME,
                ISSUEDT,
                TITLE,
                DOCTYPE,
                VALIDITY
        FROM DOCUMENT
        WHERE   DRN > &FIRST_DOC_TO_PRINT
      AND   DRN <= &LAST_DOC_TO_PRINT
..

.DEFINE     CLASSIFICATION
      SELECT      CLASSIF_NAME
      INTO        CLASSIF_NAME
      FROM        CLASSIFICATION
      WHERE       CLASSIF =
```

```
                        (SELECT CLASSIF
                        FROM  DOCUMENT
                        WHERE DRN = &DRN)
..
.REM        DEFINE PROCEDURAL MACROS
.REM
.DEFINE     BODY
.EXECUTE CLASSIFICATION
#SP 1
      #T 1 #S 1
      #CEN
      .PRINT CLASSIF_NAME
      #
#S 2
      #CEN
      B R A Z I L I A N \ \ A I R \ \ F O R C E \ \ M I N I S T R Y
      #
#S 1
      #CEN PROCESSING SHEET #
#S 2
#TE
#T 2
      Prot #:
      .PRINT P_SC
      #NC
      -
      #NC
      .PRINT P_OC
      #NC
      /
      #NC
      .PRINT P_NUMBER
      #NC /
      #NC
      .PRINT P_YEAR
      #NC
      DRN:
      .PRINT CDRN
      -
      .PRINT VERIFIER
#S 1 #TE #T 1
      .PRINT DOCTYPE
      .PRINT CLASSIF_NAME
      from
      .PRINT ORGNAME
      ID #
      .PRINT IDNR
      /
      .PRINT IDCOMPL
      of
      .PRINT ISSUEDT
#S 1  #TE #T 4
```

```
      Validity:
      .PRINT VALIDITY
      #NC
      Title:
      #NC
      .PRINT TITLE
#S 1
#TE #T 1
```

```
#TE
#T 5 FROM #NC ¦ #NC TO #NC ¦ #NC DISPATCH #N
#TE #T 6
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
¦ #NC  ¦ #NC
#S 1 #TE #T 1
      #CEN
```

```
      .PRINT CLASSIF_NAME
      #
#NP #TE
..
.REM
.REM                    PROCEDURE SECTION
.REM
.EXECUTE FIRST
.EXECUTE LAST
.REPORT DOCUMENT BODY
.STOP
```

Processing Sheet Sample

CONFIDENTIAL

B R A Z I L I A N A I R F O R C E M I N I S T R Y

PROCESSING SHEET

Prot #: 21-23/345 /89 DRN: 89100001 - 8

BOL CONFIDENTIAL from BAAN ID # 221 / IDFR/89 of 23-FEB-89

Validity: 16-FEB-94 Title: THIS IS THE TITLE OF DOCUMENT 89100001

FROM	TO	DISPATCH

CONFIDENTIAL

File A44_3.RPT, Print Thesaurus Terms by Hierarchical Relationship

```
.REM FILE NAME:  A44_3.RPT
.REM PARENT MODULE NAME: MAINTAIN THESAURUS
.REM MODULE NAME: PRINT TERMS
.REM REPORT NAME: ORDER BY TERM HIERARCHICAL RELATIONSHIP


.REM        DEFINE TAB STOPS ON THE REPORT
#DT 1   08 73 #
#DT 2   08 27  66 73  #
#DT 3   11 18  20 73  #
#DT 4   08 73  #
#T 1
#PAGE 1 90

.REM                  DECLARE VARIABLES

.DECLARE TODAY         EDATE
.SET     TODAY         $$DATE$$
.DECLARE TIME_OF_REP   A5
.SET     TIME_OF_REP   $$TIME$$

.DECLARE CLASSCD       A2
.DECLARE CLASS         A20
.DECLARE TERM          A50

.REM ****  VARIABLES USED TO BREAK PAGES
.DECLARE COUNTER       99
.SET     COUNTER       6
.DECLARE PAGE_NO       99
.SET     PAGE_NO       1
.DECLARE MAX_LINES     99
.SET     MAX_LINES     62

.REM ****  VARIABLES USED TO BREAK CLASS NAME
.DECLARE SAVE_CLASSCD A2
.SET SAVE_CLASSCD '**'

.REM                              SELECT MACROS

.DEFINE           SEL_TERM

   SELECT LPAD(' ',3*LEVEL) || LEVEL ||'. '|| MAINTR
   INTO   TERM
   FROM RELSHIPS
   CONNECT BY PRIOR MAINTR = SECTR AND REL = 'BT'
   START WITH MAINTR = 'ROOT'
   AND REL='BT'
   AND SECTR IS NULL
..
```

```
.REM        DEFINE PROCEDURAL MACROS

.DEFINE              BREAK_PAGE
     .&label1
        .IF &COUNTER<&MAX_LINES THEN label2
        .SET COUNTER 6
        .REM the heading has 5 lines
        .ADD PAGE_NO PAGE_NO 1
        .NP
        .HEAD
     .&label2
        .ADD COUNTER COUNTER 1
..


.DEFINE              HEAD

#TE
#T 1
      #CEN
      THESAURUS REPORT A44_3 - ORDER BY TERM HIERARCHICAL ASSOCIATION
      #
#TE
#T 2
      .PRINT TODAY
      .PRINT TIME_OF_REP
      #NC
      PAGE
      .PRINT PAGE_NO
#TE
#S 1
#T 3
      Level
      #NC
      Term
#S 1
      .BODY
..


.DEFINE                BODY

#TE
#T 4
      .BREAK_PAGE
      #CL
         .FPRINT TERM
      #
         #NC
..
```

```
.DEFINE                  FOOT
#TE
#S 3
#CUL  END OF REPORT #
..
.REM ******************************************************
.REM                   PROCEDURE SECTION

.REPORT SEL_TERM BODY HEAD FOOT
.STOP
```

File A44_4.RPT, Print Thesaurus Terms Giving all Relationships

```
.REM FILE NAME:  A44_4.RPT
.REM PARENT MODULE NAME: MAINTAIN THESAURUS
.REM MODULE NAME: PRINT TERMS
.REM REPORT NAME: ORDER BY TERM AND RELATIONSHIPS


.REM        DEFINE TAB STOPS (TABLES) ON THE REPORT
#DT 1   08 73 #
#DT 2   08 27  66 73  #
#DT 3   08 33  35 73  #
#DT 4   35 43  45 73  #
#DT 5   11 73  #
#T 1
#PAGE 1 90

.REM                  DECLARE VARIABLES

.DECLARE TODAY         EDATE
.SET     TODAY         $$DATE$$
.DECLARE TIME_OF_REP   A5
.SET     TIME_OF_REP   $$TIME$$

.DECLARE CLASSCD       A2
.DECLARE CLASS         A20
.DECLARE TERM          A25
.DECLARE REL           A2
.DECLARE RELNAME       A8
.DECLARE SECTERM       A25
.DECLARE NOTE         A60

.REM ****  VARIABLES USED TO BREAK PAGES
.DECLARE COUNTER       99
.SET     COUNTER       9
.DECLARE PAGE_NO       99
.SET     PAGE_NO       1
.DECLARE MAX_LINES     99
.SET     MAX_LINES     62
```

```
.REM                                    SELECT MACROS

.DEFINE             SEL_TERM

      SELECT CLASSCD, TERM
      INTO   CLASSCD, TERM
      FROM   TERMS
      ORDER BY TERM, CLASSCD
..

.DEFINE             SEL_CLASS
      SELECT CLASS
      INTO   CLASS
      FROM   CLASSES
      WHERE  CLASSCD = &CLASSCD
..

.DEFINE             SEL_RELATIONSHIPS
      SELECT        REL, SECTR
      INTO          REL, SECTERM
      FROM          RELSHIPS
      WHERE         MAINTR = &TERM
      ORDER BY REL, SECTR
..

.DEFINE             SEL_RELNAME
      SELECT        RNAME
      INTO          RELNAME
      FROM          RELNAME
      WHERE         REL = &REL
..

.DEFINE             SEL_NOTES
      SELECT        NOTE
      INTO          NOTE
      FROM          NOTES
      WHERE         TERM = &TERM
..

.REM          DEFINE PROCEDURAL MACROS

.DEFINE  BREAK_PAGE
     .&label1
        .IF &COUNTER<&MAX_LINES THEN label2
        .SET COUNTER 9
        .REM set counter with the number of lines in the header
        .REM plus the top and bottom margins
        .ADD PAGE_NO PAGE_NO 1
        .NP
        .HEAD1
```

```
     .&label2
        .ADD COUNTER COUNTER 1
..

.DEFINE      SEL_RELATIONSHIPS_BODY
    #TE
    #T 4
    .EXECUTE SEL_RELNAME
    .PRINT RELNAME
    #NC
    .PRINT SECTERM
    #NC
    .BREAK_PAGE
..

.DEFINE      SEL_NOTES_BODY
    #TE
    #T 5
    .PRINT NOTE
    #NC
    .BREAK_PAGE
..


.DEFINE  HEAD1

#TE
#T 1
      #CEN
      THESAURUS REPORT A44_4 - ORDER BY TERM AND RELATIONSHIPS
      #
#TE
#T 2
      .PRINT TODAY
      .PRINT TIME_OF_REP
      #NC
      PAGE
      .PRINT PAGE_NO
#TE
#S 1
#T 3
      Term
      #NC
      Class Number & Class Name
      #NC
      \  \ \Scope Note
      #NC
      Relation\  \Secondary Term
#S 1
..

.DEFINE                  HEAD2
```

```
      .HEAD1
      .BODY
..


.DEFINE                  BODY

.EXECUTE SEL_CLASS
#TE
#T 3
         .PRINT TERM
         #NC
         .PRINT CLASSCD
         .PRINT CLASS
         #NC
         .REM the following nested report gets the relationships
         .REPORT SEL_NOTES          SEL_NOTES_BODY
         .REPORT SEL_RELATIONSHIPS SEL_RELATIONSHIPS_BODY
         .BREAK_PAGE
..

.DEFINE     FOOT
#TE
#S 3
#CUL  END OF REPORT #
..

.REM
.REM                    PROCEDURE SECTION
.REM
.REPORT SEL_TERM BODY HEAD2 FOOT
.STOP
```

Bibliography

Chen, Peter P. "The Entity-Relational Model: Toward a Unified View of Data," ACM Transactions on Database Systems, 1 #1: 9-36 (January 1976).

Codd, E. F. "A Relational Model for Large Shared Data Banks," Communications of the ACM, 13 #6: 377-387 (June 1970).

Horowitz, Ellis and Sahni, Sartaj. Fundamentals of Data Structures in Pascal. Rockville, MA: Computer Science Press, 1987.

Jackson, D. M. "Classification, Relevance, and Information Retrieval." Advances in Computers. 59-125. New York: Academic Press, 1971.

Korth, Henry F. and Silberschatz, Abraham. Database System Concepts. New York: McGraw-Hill Book Company, 1986.

Lancaster, F. W. Vocabulary Control for Information Retrieval. Arlington: Information Resources Press, 1986.

Lans, Rick F. Introduction to SQL. New York: Addison-Wesley Publishing Company, 1988.

Ministerio da Aeronautica. Instrucoes sobre Correspondencia e Atos Oficiais do Ministerio da Aeronautica, ICAER. IMA 10-01. Guaratingueta, Brazil: edited by the Escola de Especialistas da Aeronautica, 1976.

Ministerio da Aeronautica. Regulamento para Salvaguarda de Assuntos Sigilosos, RSAS. Brasilia, Brazil: edited by the Central Agency of the National Information Service, 07 JAN 1977.

Pressman, Roger S. Software Engineering, a Practitioner's Approach. New York: McGraw-Hill Book Company, 1987.

Oracle Corporation. Oracle for IBM PC/MS-DOS Installation and User's Guide. Part Number 1022-V5.1A. Belmont, CA: edited by Oracle Corporation, 1987.

Zussman, John Unger and others. SQL*Forms, Designers's Reference, Version 2.0. Part Number 3304-V2.0. Belmont, CA: edited by Oracle Corporation, 1987.

Simon and Schuster. Webster's New World Dictionary of the American Language. Cleveland: New World Dictionaries, 1984.

Stonebraker, Michael. Readings in Database Systems. San Mateo, CA: Morgan Kaufmann Publishers, INC, 1988.

Vita

Lieutenant Colonel Antonio F. Bernardo Silva [redacted] He joined the Brazilian Air Force in 1966 and graduated in 1971 with a Bachelor of Science (B.S.) degree from the Brazilian Air Force Academy; at the same time he completed his pilot training and received his wings. Upon graduation, he was designated to the Fighter Pilot Selection Course, receiving his Fighter Pilot Wings in 1972, as the more efficient pilot of his class. He then served as a F5-E pilot in the 1st Grupo de Aviacao de Caca, Santa Cruz, Rio de Janeiro up to his designation to the Air Defense Command, Brasilia, in 1978. His first contact with computers were that same year, in the "Air Defense and Air Traffic Control Integrated Center", Brasilia, when he was assigned to a training course as Air Defense Control Chief. In 1982 he graduated from the Roman Catholic Faculty of Technology, Brasilia, in a four-quarter Specialization Course on Computer Science. In 1983 he was designated to participate in a work group to create the Air Ministry Data Processing Center, Brasilia, which became operational in 1984. He was the head of the Center until entering the School of Engineering, Air Force Institute of Technology, in June 1987. During his assignment at that Center he developed several systems, mainly oriented to database systems.

[redacted]