

Air Force Institute of Technology

AFIT Scholar

Theses and Dissertations

Student Graduate Works

3-2002

Application of Information Retrieval Techniques to Heterogeneous Databases in the Virtual Distributed Laboratory

Rodney D. Lykins

Follow this and additional works at: <https://scholar.afit.edu/etd>



Part of the [Databases and Information Systems Commons](#)

Recommended Citation

Lykins, Rodney D., "Application of Information Retrieval Techniques to Heterogeneous Databases in the Virtual Distributed Laboratory" (2002). *Theses and Dissertations*. 4420.

<https://scholar.afit.edu/etd/4420>

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact richard.mansfield@afit.edu.



APPLICATION OF INFORMATION RETRIEVAL TECHNIQUES TO
HETEROGENEOUS DATABASES IN THE VIRTUAL DISTRIBUTED
LABORATORY

Rodney D. Lykins, 1st Lieutenant, USAF

AFIT/GCS/ENG/02M-06

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

Report Documentation Page

Report Date 5 Mar 02	Report Type Final	Dates Covered (from... to) Mar 2001 - Mar 2002
Title and Subtitle Application of Information Retrieval Techniques to Heterogeneous Databases in the Virtual Distributed Laboratory	Contract Number	
	Grant Number	
	Program Element Number	
Author(s) 1st Lt Rodney D. Lykins, USAF	Project Number	
	Task Number	
	Work Unit Number	
Performing Organization Name(s) and Address(es) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 P Street, Bldg 640 WPAFB OH 45433-7765	Performing Organization Report Number AFIT/GCS/ENG/02M-6	
Sponsoring/Monitoring Agency Name(s) and Address(es) AFRL?SN ATTN: Eric Baenen 5200 Springfield Pike, Suite 200 Dayton, OH 45431-1289	Sponsor/Monitor's Acronym(s)	
	Sponsor/Monitor's Report Number(s)	
Distribution/Availability Statement Approved for public release, distribution unlimited		
Supplementary Notes The original document contains color images.		

Abstract

The Department of Defense (DoD) maintains thousands of Synthetic Aperture Radar (SAR), Infrared (IR), Hyper-Spectral intelligence imagery and Electro-Optical (EO) target signature data. These images are essential to evaluating and testing individual algorithm methodologies and development techniques within the Automatic Target Recognition (ATR) community. The Air Force Research Laboratory Sensors Directorate (AFRL/SN) has proposed the Virtual Distributed Laboratory (VDL) to maintain a central collection of the associated imagery metadata and a query mechanism to retrieve the desired imagery. All imagery metadata is stored in relational database format for access from agencies throughout the federal government and large civilian universities. Each set of imagery is independently maintained at each agency's location along with a local copy of the associated metadata that is periodically updated and sent to the VDL. This research focuses on applying information retrieval techniques to the multiple heterogeneous imagery metadata databases to present users the most relevant images based on user defined search criteria. More specifically, it defines a hierarchical concept thesaurus development methodology to handle the complexities of heterogeneous databases and the application of two classic information retrieval models. The results indicate this type of thesaurus-based approach can significantly increase the precision and recall levels of retrieving relevant documents.

Subject Terms

Virtual Distributed Laboratory, Information Retrieval, Hierarchical Thesaurus, Vector Model, Extended Boolean Model.

Report Classification

unclassified

Classification of this page

unclassified

Classification of Abstract

unclassified

Limitation of Abstract

UU

Number of Pages

93

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or United States Government.

APPLICATION OF INFORMATION RETRIEVAL TECHNIQUES TO
HETEROGENEOUS DATABASES IN THE VIRTUAL DISTRIBUTED
LABORATORY

Presented to the faculty of the Graduate School of Engineering & Management
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the
Requirements for the Degree of
Master of Science (Computer Science)

Rodney D. Lykins, B. S.

1st Lieutenant, USAF

March 2002

APPROVED FOR PUBLIC RELEASE, DISTRIBUTION UNLIMITED

APPLICATION OF INFORMATION RETRIEVAL
TECHNIQUES TO HETEROGENEOUS DATABASES IN THE
VIRTUAL DISTRIBUTED LABORATORY

Rodney D. Lykins, B. S.
1st Lieutenant, USAF

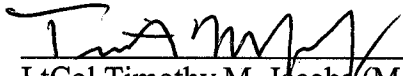
Approved:



Major Karl S. Mathias (Chairman)

5 MAR 02

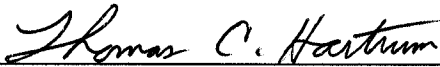
date



LtCol Timothy M. Jacobs (Member)

5 Mar 02

date



Dr. Thomas C. Hartrum (Member)

5 Mar 02

date

Acknowledgments

This information retrieval research was only possible with the help and support of many different people. I would like to thank God for allowing me to be here and have the support and dedication to complete this monumental task. I would like to thank my wife for her unwavering support during the past eighteen months. Without her loving support and understanding during the trials of the last year none of this would have been possible. I would also like to thank my advisor, Maj Karl Mathias, for providing assistance and guidance when needed. I would also like to thank my sponsor, Eric Baenen from AFRL/SN, for providing me such an interesting and challenging thesis topic to work on. I would also like to thank the other members of my committee, LtCol Tim Jacobs and Dr. Thomas Hartrum. Finally, I would like to thank my fellow classmates who kept me sane and focused on the task at hand.

Table of Contents

	Page
Acknowledgments	iv
Table of Figures	viii
I. Introduction.....	1
1.1 Definition of Terms.....	2
1.2 VDL Problem Statement.....	2
1.3 VDL Research Focus	6
1.4 Summary	7
II. Literature Review	8
2.1 Introduction.....	8
2.2 VDL Overview.....	9
2.3 Metadata.....	10
2.4 Relational Database Model	12
2.4.1 Data Storage.....	12
2.4.2 Data Access and Manipulation	14
2.5 Heterogeneous Databases	15
2.5.1 Schema Differences	15
2.5.2 Identical Data With Different Name.....	15
2.5.3 Different Data With Same Name	16
2.5.4 Unit Of Measure Differences.....	16
2.6 Information Retrieval.....	17
2.6.1 Information Retrieval Models.....	17
2.6.1.1 Boolean Model	18
2.6.1.2 Vector Model.....	19
2.6.1.3 Extended Boolean Model	20
2.6.2 Information Retrieval Techniques	21
2.6.2.1 Inverted File.....	21
2.6.2.2 Stop Words	23
2.6.2.3 Stemming.....	23
2.6.2.4 Thesauri	24
2.7 B-Trees.....	26
2.8 Summary	26
III. Methodology	28
3.1 Introduction.....	28
3.2 Development Tools.....	29
3.2.1 Call and Return Architecture	29
3.2.2 Programming Language.....	29

	Page
3.3 Hierarchical Thesaurus and Indexing Tool.....	30
3.3.1 General Approach.....	30
3.3.2 Document Representation.....	31
3.3.3 Hierarchical Thesaurus.....	31
3.3.4 Database Parsing.....	33
3.3.5 Indexing.....	34
3.3.5.1 Document Indexing.....	35
3.3.5.2 Column Headings.....	36
3.4 Query Tool.....	36
3.4.1 General Approach.....	36
3.4.2 Relevant Document Retrieval.....	36
3.4.3 Relevant Document Ranking.....	38
3.5 Design Architecture.....	38
3.5.1 Schema Integration Tool.....	39
3.5.1.1 Parsing.....	40
3.5.1.2 Hierarchy Creation.....	40
3.5.1.3 Indexing.....	41
3.5.2 Query Tool.....	42
3.5.2.1 Filtering.....	43
3.5.2.2 Relevant Document Ranking.....	44
3.6 Summary.....	45
IV. Implementation and Results.....	46
4.1 Introduction.....	46
4.2 Design Issues.....	46
4.2.1 Memory.....	46
4.2.2 Speed.....	47
4.3 Hierarchical Thesaurus and Indexing Tool.....	48
4.3.1 Parsing Documents.....	48
4.3.2 Hierarchy Thesaurus Generation.....	49
4.3.3 Indexing.....	49
4.4 Query Tool.....	52
4.4.1 Relevant Image Filtering.....	52
4.4.2 Relevant Image Ranking.....	53
4.4.2.1 Vector Model.....	53
4.4.2.2 Extended Boolean Model.....	55
4.5 Tool Demonstration.....	55
4.5.1 Hardware and Software Platforms.....	56
4.5.2 Databases.....	56
4.5.2.1 Hierarchical Thesaurus Testing Databases.....	56
4.5.2.2 Queries and Relevant Documents.....	57
4.5.3 Evaluation Criteria.....	57
4.5.3.1 Indexing Speed.....	58
4.5.3.2 Query Speed.....	58

	Page
4.5.3.3 Relevant Document Retrieval.....	59
4.5.3.4 Recall / Precision Variable Explanation.....	59
4.5.3.5 Precision vs. Recall.....	60
4.5.4 Results.....	62
4.5.4.1 Indexing Speed	62
4.5.4.2 Query Speed	62
4.5.4.3 Relevant Document Retrieval.....	64
4.6 Summary	67
V. Conclusions and Future Work.....	68
5.1 Summary of Research	68
5.2 Results.....	69
5.3 Future Research Recommendations.....	69
5.3.1 Relevance Feedback	69
5.3.2 Phrase to Phrase and Phrase to Word Mapping.....	70
5.3.3 Web-based Application for Multiple Users.....	70
5.3.4 Increased Indexing and Query Retrieval Speed.....	70
5.4 Summary	71
A. Relevant Documents Retrieved.....	72
A.1 Query 1.....	72
A.2 Query 2.....	72
A.3 Query 3.....	72
A.4 Query 4.....	73
B. Query Speed	75
B.1 Query Speed for 10,000 Documents	75
B.2 Query Speed for 20,000 Documents	75
B.3 Query Speed for 30,000 Documents	76
B.4 Query Speed for 40,000 Documents	76
B.5 Query Speed for 50,000 Documents	76
B.6 Query Speed for 60,000 Documents	77
B.7 Query Speed for 70,000 Documents	77
B.8 Query Speed for 80,000 Documents	77
B.9 Query Speed for 90,000 Documents	78
B.10 Summation of Query Speeds.....	78
Bibliography.....	79
Vita.....	81

List of Figures

	Page
Figure 1. VDL Metadata Query Environment	3
Figure 2. Query Tool 1.0 User Interface	5
Figure 3. Query Tool 2.0 User Interface	5
Figure 4. Virtual Distributed Laboratory Architecture	9
Figure 5. Example of a SysTables System Table.....	12
Figure 6. Partial VDL Metadata Record entry	12
Figure 7. Conceptual View of an Inverted File.....	22
Figure 8. A Partial Hierarchy of Aircraft Words and Categories.....	25
Figure 9. B-Tree Logical View	26
Figure 10. Hierarchical Thesaurus Example	32
Figure 11. Pipeline Architecture Between the Two Tools.....	38
Figure 12. Available Method Calls of the Schema Integration Tool	39
Figure 13. Schema Integration Tool Method Call Chart.....	40
Figure 14. Preventing Category Cyclical Graph Example.....	41
Figure 15. Available Method Calls of the Query Tool	43
Figure 16. Query Tool Method Call Chart.....	43
Figure 17. Column Word Architecture	50
Figure 18. Word Architecture	50
Figure 19. Created Hierarchies of Synonyms and Concepts for Testing.....	57
Figure 20. Variable Explanation for Precision and Recall.....	59

	Page
Figure 21. Precision-Recall Graph for Example	61
Figure 22. Indexing Speed with Various Levels of Caches	62
Figure 23. Query Speed of Various IR Models.....	63
Figure 24. Precision-Recall Graph for Query 1	65
Figure 25. Precision-Recall Graph for Query 2	66
Figure 26. Precision-Recall Graph for Query 3	66
Figure 27. Precision-Recall Graph for Query 4	67

Abstract

The Department of Defense (DoD) maintains thousands of Synthetic Aperture Radar (SAR), Infrared (IR), Hyper-Spectral intelligence imagery and Electro-Optical (EO) target signature data. These images are essential to evaluating and testing individual algorithm methodologies and development techniques within the Automatic Target Recognition (ATR) community. The Air Force Research Laboratory Sensors Directorate (AFRL/SN) has proposed the Virtual Distributed Laboratory (VDL) to maintain a central collection of the associated imagery metadata and a query mechanism to retrieve the desired imagery. All imagery metadata is stored in relational database format for access from agencies throughout the federal government and large civilian universities. Each set of imagery is independently maintained at each agency's location along with a local copy of the associated metadata that is periodically updated and sent to the VDL. This research focuses on applying information retrieval techniques to the multiple heterogeneous imagery metadata databases to present users the most relevant images based on user defined search criteria. More specifically, it defines a hierarchical concept thesaurus development methodology to handle the complexities of heterogeneous databases and the application of two classic information retrieval models. The results indicate this type of thesaurus-based approach can significantly increase the precision and recall levels of retrieving relevant documents.

APPLICATION OF INFORMATION RETRIEVAL TECHNIQUES TO HETEROGENEOUS DATABASES IN THE VIRTUAL DISTRIBUTED LABORATORY

I. Introduction

The Department of Defense (DoD) maintains thousands of Synthetic Aperture Radar (SAR), Infrared (IR), Hyper-Spectral intelligence imagery and Electro-Optical (EO) target signature data. These images are essential to evaluating and testing individual algorithm methodologies and development techniques within the Automatic Target Recognition (ATR) community. The Air Force Research Laboratory Sensors Directorate (AFRL/SN) has proposed the Virtual Distributed Laboratory (VDL) to maintain a central collection of the associated imagery metadata and a query mechanism to retrieve the desired imagery. All imagery metadata is stored in relational database format for access from agencies throughout the federal government and large civilian universities. Each set of imagery is independently maintained at each agency's location along with a local copy of the associated metadata that is periodically updated and sent to the VDL.

Previous research [Ward, 00] in VDL focused on the integration of user profiling and user interface analysis to increase user productivity by decreasing query input time. It proposed improvements to the latest interface based on objective interface evaluation

criteria and accomplished user profiling by incorporating a server-side implementation of user defined profiles. Additional research [Hooten, 01] conducted performance analysis on bandwidth characterization issues. The analysis tried to determine an optimal configuration of servers and images based on the current network configuration to reduce bandwidth. However, this research effort was hindered by the limitations of the selected network-modeling tool. This current research focuses on applying information retrieval techniques to the multiple heterogeneous imagery metadata databases to present users the most relevant images based on user defined search criteria. More specifically, it defines a thesaurus development methodology to handle the complexities of heterogeneous databases and the application of two classic information retrieval models.

1.1 Definition of Terms

In order to provide a better understanding of this research effort, a definition of key terms as used in the context of this document are as follows:

- **Database:** Any relational database, e.g. Access, MySQL, SQL Server, etc
- **Document:** A record within a database consisting of entries of metadata, corresponding to an individual image
- **Model:** Any information retrieval model, e.g. Boolean, Vector, Extended Boolean, etc

1.2 VDL Problem Statement

The mission of the VDL is to facilitate cooperative research, development, and algorithm evaluation by providing communications, information services, and information retrieval for the entire ATR community. To this end the Air Force Research

Laboratory Sensors Directorate has consolidated several heterogeneous imagery metadata databases into a single location as part of the VDL environment. The VDL environment, as seen in Figure 1, establishes a central library for metadata databases while allowing participating agencies to retain local control over their images.

With the imagery repositories occupying over twenty terabytes, containing over 750,000 images, and the metadata databases currently over three gigabytes in size and growing, a primary goal of AFRL/SN is to provide an effective means to locate and retrieve images based on queries of associated metadata. Current query tool implementations, discussed later, may return thousands of documents with no mechanism for the user to distinguish between relevant documents (metadata records) and non-relevant documents. This makes the user's job of selecting images to evaluate ATR algorithms very difficult. This research discusses the theory and implementation behind a new set of VDL tools that normalize differing schemas of heterogeneous databases while allowing a user to submit a single query to multiple databases and retrieve ranked relevant documents using classic information retrieval techniques.

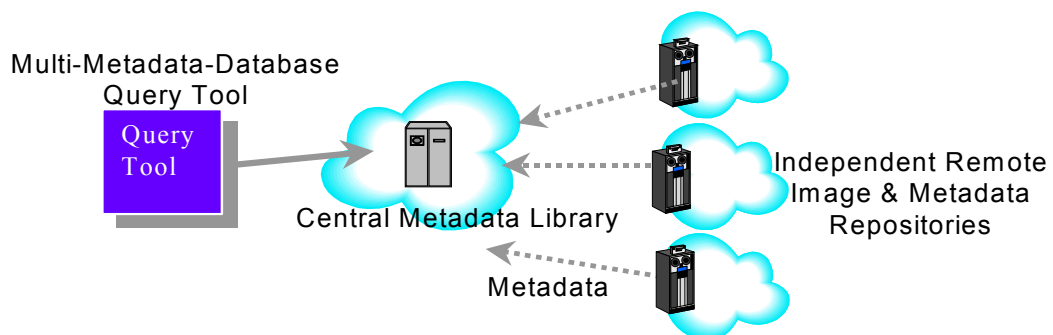


Figure 1. VDL Metadata Query Environment

Previous implementations of query tools by AFRL/SN have met with limited success. This is because of the inability to interface with more than one database at a time due to the differing naming conventions for column headings and data values among the databases. In addition, no ranking mechanism to distinguish between the most relevant and least relevant documents was provided. The first query tool, AQT 1.0, was hard-coded to work directly with the metadata database held at Wright-Patterson AFB, OH. Programming to a specific database makes it impossible to locate relevant documents from any other database within VDL without modifying the existing code. The next query tool, AQT 2.0 improved upon its predecessor by adapting itself to each database schema upon startup. This improvement made it possible to interface with different databases, however, it was still only capable of interfacing with one database at a time. To search multiple databases, the user must start the tool and submit a query, direct the tool to another database and restart the tool to submit additional queries. In both implementations, each tool submitted queries via SQL thereby eliminating any mechanisms of ranking documents by relevance to the user's query. In addition, the interfaces presented users with hundreds of various combinations of searchable image characteristics. The user interfaces of AQT 1.0 and AQT 2.0 can be seen in Figure 2 and Figure 3, respectively.

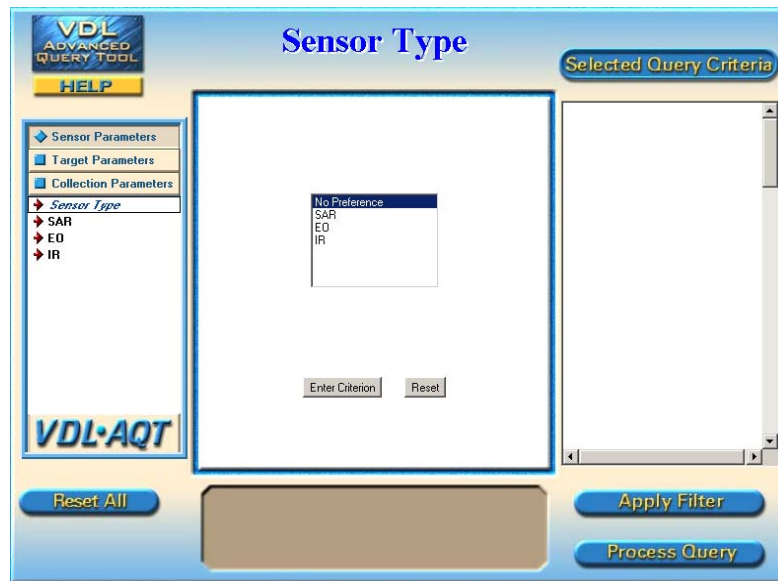


Figure 2. Query Tool 1.0 User Interface

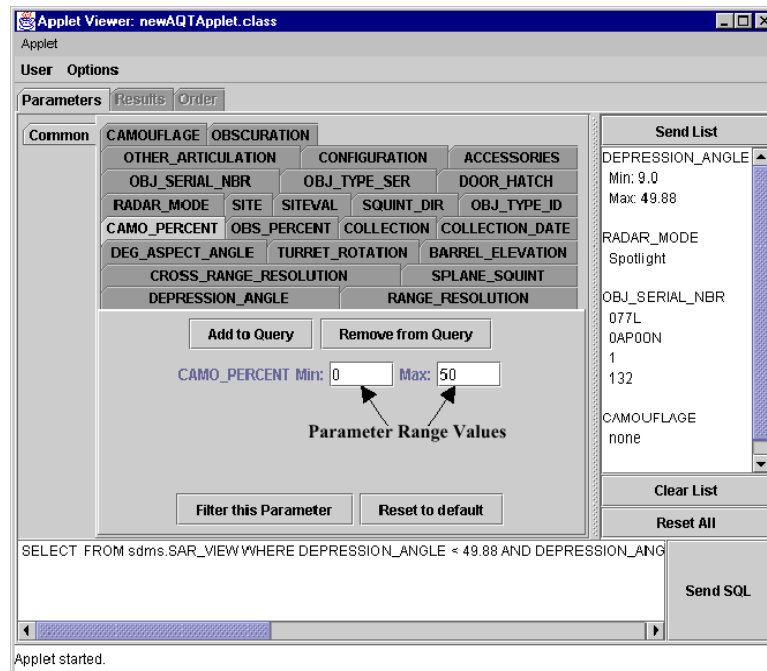


Figure 3. Query Tool 2.0 User Interface

Both of these tools can be powerful if the user knows where to look in the interface and exactly what they are looking for. However, due to the myriad of selections for the user to make, the interface complexities make the tool tedious to navigate and cumbersome to use. These tools make each query a mundane exercise to narrow the search to images with a specific set of characteristics. In addition, neither tool is robust enough to solve the inherent difficulty of VDL's proposed design of querying multiple heterogeneous databases.

1.3 VDL Research Focus

The design, development, and evaluation of a new set of tools that categorize database specific terms, column headings and data values, by assigning synonyms and is capable of accepting a single user query, evaluating it across multiple heterogeneous databases, and returning the most relevant documents to the user is the primary focus of this research. For this research, the developed tools responsibilities include:

- Schema Integration Tool
 - Working in a heterogeneous database environment
 - Provide means to create a thesaurus through the creation of hierarchical categories and synonyms
 - Indexing documents according to information retrieval techniques
- Query Tool
 - Presenting the user an easily understood interface for query submission and results viewing
 - Incorporate synonyms and categories through local query expansion

- Identifying and ranking relevant documents according to existing information retrieval models for presentation to the user

1.4 Summary

The Virtual Distributed Laboratory contains large databases of searchable imagery metadata for evaluating and testing automated target recognition algorithms. While the existing user interface can adapt itself to each individual database for relevant images, it is unable to search multiple heterogeneous databases at the same time with one user query. This new research evaluates techniques for the retrieval of relevant images across multiple heterogeneous databases. Specifically, the primary focus of this research is the application of a thesaurus-based approach for schema and data normalization for multiple heterogeneous databases and the evaluation of two classic information retrieval models, the Vector Model and Extended Boolean Model, as applied to the VDL environment.

Chapter II of this paper provides an overview of the VDL concept, an overview of relational database theory and schema/data integration and normalization, an introduction to information retrieval concepts and the information retrieval models used in this research. Chapter III explains the methodology used to develop the schema/data normalization tool and the multi-database query tool. Chapter IV outlines the design and implementation of each of these tools. Chapter V features the test cases and results, analysis of results, conclusions and suggestions for future research.

II. Literature Review

2.1 Introduction

The VDL concept encompasses several different technologies that are central to understanding the complexities of retrieving relevant images from multiple heterogeneous databases. Since this research primarily involves the Virtual Distributed Laboratory, an overview of the VDL and its architecture is discussed in this chapter. The role of metadata and how it can be used within the information retrieval domain is also analyzed. Since the metadata resides in relational databases, relational database theory is discussed along with Structured Query Language or SQL. This research places great emphasis on multiple co-located heterogeneous databases, therefore, the complexities and difficulties of working in that environment are presented. In addition, since the primary focus of this research effort involves the application and evaluation of information retrieval models to retrieve and rank relevant documents, the role information retrieval concepts and models play is vitally important and is therefore covered in this literature review. Three information retrieval models, the Boolean, Extended Boolean, and Vector Models are discussed in detail to provide a better understanding of the differing approaches to ranking relevant documents. Since the word synonyms and concept hierarchy must be maintained in persistence storage, a disk-based B+ Tree data structure is explored to discover its advantages and disadvantages and is the final topic in this literature review.

2.2 VDL Overview

The Virtual Distributed Laboratory (VDL) is a DoD wide Distributed Collaborative Development Environment (DCDE) established to facilitate cooperative research and development within the automatic target recognition community [VDL Marketing Slides]. It is sponsored by the Office of the Under Secretary of Defense for Acquisition and Technology (OSD ACQ) and was created as an outgrowth of the Defense Advanced Research Projects Agency's (DARPA's) Moving and Stationary Target Acquisition and Recognition (MSTAR) program as discussed in [DARPA Website]. The VDL architecture consists of five major components as seen in Figure 4 [VDL White Paper].

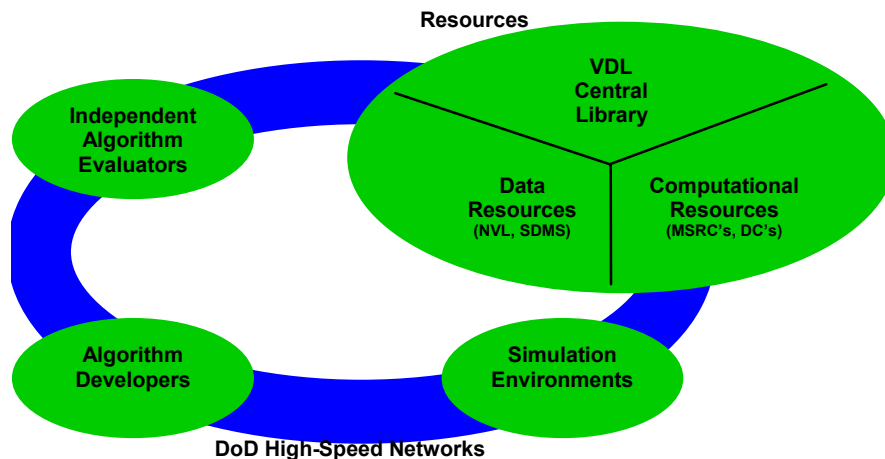


Figure 4. Virtual Distributed Laboratory Architecture

AFRL/SN was funded to create the On-Line central library which has grown to consist of four major elements:

- The web-based on-line library
- Signature and Imagery Data locating tools

- Topic restricted search engines
- Synchronous and asynchronous collaborative tools

Allowing users to search the imagery metadata repositories based on defined criteria is the primary purpose of the signature and imagery data locating tools. Locating candidate signature and imagery data is critical to the success of developing good ATR algorithms due to the large amounts of images required to train, test, and evaluate these algorithms.

Signature files and imagery data are stored in large repositories independently maintained by various agencies throughout the DoD. A description of each file and image, metadata, is stored separately from the data itself in comma separated value (.csv) files that are periodically updated to the VDL central library. These .csv files are imported into a relational database program (i.e. Access, MySQL, SQL Server, etc). For this research Microsoft's Access relational database program was used. The role of metadata in this research and relational DBMS models are discussed in the next two sections.

2.3 Metadata

Metadata is essentially data about the data. [Dempsey, 1997] provides a more formal definition of metadata:

“Metadata is data associated with objects which relieves their potential users of having to have full advance knowledge of their existence or characteristics”

For example, in a relational DBMS the schema identifies some of the metadata such as the name of the relations, the fields or attributes of each relation, the domain of each attribute, etc. This information is stored in special tables called system tables. Figure 5 provides an example of a system table in a relational DBMS. This table contains a record of each table present in the database. These records store the number of columns of each table and each table's primary key.

In general, metadata can be broken down into five different categories depending on its functionality [Gilliland]:

- **Administrative:** Used in managing and administering information resources
- **Descriptive:** Used to describe or identify information resources
- **Preservation:** Related to the preservation management of information resources
- **Technical:** Related to the level and type of use of information resources
- **Use:** Related to the level and type of use of information resources

The administrative and descriptive definitions of metadata most closely identify the role of metadata as it is used in the VDL environment. The metadata contained about each image not only contains a description of the image, but also characteristics as to how it was acquired. Figure 6 displays a partial metadata record entry in the VDL environment taken from the Sensors Data Management System (SDMS) database at Wright-Patterson AFB, OH.

In this example, the metadata identifies a number of defining characteristics that describe this image:

- **Image size:** 22896000
- **Date image was taken:** 27 April 1998
- **Location image was taken:** Eglin AFB, FL
- **Primary weapon system in image:** T-72 Russian Tank
- **Specific serial number of weapon system:** A10

Since the metadata is imported into a relational DBMS from its .csv file format, an overview of some basic principles behind the relational DBMS Model are discussed next.

TABLE NAME	NUMBER OF COLUMNS	PRIMARY KEY
Eo_view	26	ID
Ir_view	26	ID
Sar_view	32	ID

Figure 5. Example of a SysTables System Table

FILESIZE	COLLECTION_DATE	SITE	OBJ_TYPE_ID	OBJ_SERIAL_NBR
228960000	19980427	eglin_fl	T72	A10

Figure 6. Partial VDL Metadata Record entry

2.4 Relational Database Model

2.4.1 Data Storage

The Relational Database Model is the most widely used data model in the marketplace. In the late 1960s at IBM Research, Dr. E.F. Codd established the foundation for relational database theory. The Relational Model is based on the concept of a collection of tables in which all data is stored. These tables represent data as a

collection of relations where columns are attributes and rows represent entities [Codd, 1970]. Each column name in a table must be unique and all attributes of an entity, when taken together, represent a “key” that uniquely identifies that entity.

A RDBMS is based on the mathematical notions of Relational Algebra and Relational Calculus. Relational Algebra provides a collection of operations to manipulate queries through the use of set operations (union, intersection, etc.) and pure database operations (select, join, etc.). A Relational Calculus is a formal query language that allows users to simply write a single declarative expression instead of having to write a sequence of relational algebra operations.

A RDBMS can have varying degrees of performance based on its level of normalization. Database normalization is the process of efficiently organizing data in a database. The two goals of this process are to eliminate redundant data and ensure data dependencies make sense [Silberschatz, 02]. In essence, a higher order of normalization will reduce the amount of storage a database consumes and ensure the logical storage of data. There are generally five levels of normalization accepted throughout the database community. These are referred to as normal forms and are numbered from one (the lowest form of normalization, referred to as first normal form or 1NF) through five (fifth normal form or 5NF). Below are guidelines to achieve a particular normal form for a given database. Since most applications do not need to be normalized beyond third normal form, the requirements for fourth normal form and fifth normal form are not shown.

- **First Normal Form:** Eliminate duplicate columns from the same table as well as create separate tables for each group of related data and identify each row with a primary key.
- **Second Normal Form:** Remove subsets of data that apply to multiple rows of a table and place them in separate tables. Also, create relationships between these new tables and their predecessors through the use of foreign keys.
- **Third Normal Form:** Remove columns that are independent of the primary key.

2.4.2 Data Access and Manipulation

Structured Query Language (SQL) is the most common query language used to manipulate data and retrieve records contained in the relational database model. The SQL commonly referred to today was established by the American National Standards Institute (ANSI) and the International Standards Organization (ISO) in 1987 and was an outgrowth of a series of relational model query languages developed as part of the System R project [Worsley, 01]. SQL is both a Data Definition Language (DDL) and a Data Manipulation Language (DML). As a DDL, it allows an administrator or designer to define tables, create views, etc. It also provides for integrity constraints and access rights specifications. As a DML, it allows users to manipulate data and retrieve information through the use of relational algebra and relational calculus. As a consequence, in the information retrieval domain, SQL is unable to determine a degree of relevance for a user's query. This is due to its reliance of set theory and Boolean operators.

2.5 Heterogeneous Databases

The VDL On-Line Central Library of metadata is composed of multiple heterogeneous databases. A group of databases are considered to be heterogeneous if one or more of the following differences exist between them: schema differences, identical data with different name, different data with same name, and unit of measure differences. Each of these areas is discussed below.

2.5.1 Schema Differences

A database schema is essentially the logical design of the database. Entries into the database must conform to the database's schema. Typically, when working with two or more databases, their schemas will not conform to one another. For example, in one database an individual's weight may be entered as an integer, whereas in another, weight may be entered as a floating number. This can cause problems trying to retrieve data from each of the databases with the same query expression. Therefore, special procedures must be developed to retrieve data from these multiple sources.

2.5.2 Identical Data With Different Name

Having multiple databases that contain identical data with different names is another problem within heterogeneous databases. For instance, one database may contain the column value called "IMAGE_DESCRIPTION," that is used to describe the contents of an image. Yet, in another database, describing an image is placed under a column named "IMAGE_SYNOPSIS." Multiple databases may also have instances where one column contains the desired information, while another database contains the information

spread across multiple columns. In both instances, to retrieve data from these databases, some form of mapping between columns must be accomplished. This problem is not limited to column headings, it can also appear in the data values themselves.

2.5.3 Different Data With Same Name

Different data with the same name occurs when data that has different meaning appears in multiple databases having the same reference name. When dealing with information retrieval this can cause more non-relevant documents to be returned. This is because there is no manner to distinguish which data is relevant based on the index term in the query. However, this can be less of a problem, if the information retrieval system supports relevance feedback from the user as to which documents were relevant to the submitted query.

2.5.4 Unit Of Measure Differences

Unit of measure differences occur when databases determine their data values in different manners. For instance, two databases could have two identical schemas where a column named “Weight” is expecting a float value. However, Database A measures weight in pounds, where Database B measures weight in kilograms. To effectively retrieve information from both databases, some conversion formulas must be adopted to map between various databases.

2.6 Information Retrieval

Information retrieval (IR) is the study of information items in regards to their representation, storage, organization, and access [Salton, 83]. Most of the time IR deals with information in the textual domain, however, it may also involve spoken language, multimedia formats, etc. The main focus of IR is to provide the end user the most relevant information based on some given guidelines. There are two different methods of information retrieval: *ad hoc* and filtering. The *ad hoc* method is the most common form of information retrieval. An information retrieval system is deemed to be *ad hoc* if new queries are being submitted to the system while the document collection remains relatively the same. Filtering, on the hand, occurs when the queries remain relatively the same, while new documents enter the collection. In both systems, the results can be presented to the users in a ranked format based on some relevance score. However, this ranking mechanism is rarely used in a filtering system [Baeza-Yates, 1999]. Many different models and techniques can be used to return and rank relevant documents to the user. Some of these models and techniques are discussed here.

2.6.1 Information Retrieval Models

There are over a dozen information retrieval models and variations, however, only three are discussed below. Before this discussion, however, a brief discussion of the four characteristics of an information retrieval model is presented as found in [Baeza-Yates, 1999]:

- D is a set of documents in a given collection

- Q is the set of query terms from the user
- F is the framework for modeling queries, documents, and their relationships
- $R(q_i, d_j)$ is the ranking function which associates how well a document relates to a given query

Each IR model views the documents and/or queries in a different manner. In the next few sections, a detailed discussion of the most common IR model, the Boolean Model, is presented, along with the two models that were implemented in this research, the Vector Model and the Extended Boolean Model.

2.6.1.1 Boolean Model

The Boolean Model is the simplest information retrieval model where queries are specified as Boolean expressions. It uses set theory and Boolean algebra to determine whether a document is relevant or not. Queries consist of index terms separated by the words *and*, *or*, and *not*. The Boolean model looks at individual index terms and assigns a term weight of (0, 1) based on whether it appears in a document. The ranking of a document is accomplished by calculating the similarity of each document to the query based on each index term weight. A document is deemed relevant only if $sim(d_j, q) = 1$.

However, with this easy implementation and understanding comes some major drawbacks. First, the documents are either relevant or non-relevant, with no partial matches, making the comparison of relevance between documents very difficult. Next, is the difficulty to express queries as Boolean expressions because most users assume the everyday English semantics of AND and OR rather than their logical equivalence. In the

end, the Boolean approach of exact matching can either return too few or too many documents to the user.

2.6.1.2 Vector Model

The Vector model is a framework in which a degree of similarity of a document to a query can be calculated. The weighting of index terms is no longer binary, which allows partial match consideration. The degree of similarity acts as the relevance score of a document – the higher, the better.

The degree of similarity for a document is calculated by first calculating the normalized frequency of a term in a given document:

$$f_{i,j} = \frac{freq_{i,j}}{\max_i freq_{i,j}}$$

Where, $freq_{i,j}$ = raw frequency of term k_i in doc_j and $\max_i freq_{i,j}$ is the maximum freq. of all terms in doc_j

Next, calculate the inverse document frequency, idf , of term k_i :

$$idf_i = \log \frac{N}{n_i}$$

Where, N = total number of docs in the collection and n_i = number of docs index term k_i appears

Then multiply the two together to calculate the term-weight for each k_i in doc_j

$$w_{i,j} = f_{i,j} * idf_i$$

Further, the cosine of the angle between the query and each document is calculated by the following:

$$\text{sim}(d_j, q) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \frac{\sum_{i=1}^t w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^t w_{ij}^2} \times \sqrt{\sum_{i=1}^t w_{iq}^2}}$$

Where t = vector dimension

2.6.1.3 Extended Boolean Model

The Extended Boolean model is a compromise between the Boolean and Vector models. In the Extended Boolean model, query term weights are assumed to be between 0 and 1, possibly by using the following formula:

$$w_{x,j} = f_{x,j} * \frac{idf_x}{\max_i idf_i}$$

Given two index terms the $\text{sim}(q_{or}, d)$ and the $\text{sim}(q_{and}, d)$ can be calculated.

$$\text{sim}(q_{or}, d) = \sqrt{\frac{x^2 + y^2}{2}}$$

$$\text{sim}(q_{and}, d) = 1 - \sqrt{\frac{(1-x)^2 + (1-y)^2}{2}}$$

Where $x = w_{x,j}$ and $y = w_{y,j}$

Obviously, many queries have more than two key terms, therefore a generalization of the above formulas as been adopted called the *p-norm model* where $1 \leq$

$p \leq \infty$. The value of p must be defined at query time. With this in mind the query-document similarities can now be calculated as follows:

$$sim(q_{or}, d_j) = \left(\frac{x_1^p + x_2^p + \dots + x_m^p}{m} \right)^{\frac{1}{p}}$$

$$sim(q_{and}, d) = 1 - \left(\frac{(1-x_1)^p + (1-x_2)^p + \dots + (1-x_m)^p}{m} \right)^{\frac{1}{p}}$$

It can be shown that when $p = 1$, the above formulas are similar to the Vector model and when $p = \infty$, they are similar to the Boolean model. Therefore, by varying p between 1 and ∞ , the behavior of the Extended Boolean model can be more like the Vector or Boolean model respectively.

2.6.2 Information Retrieval Techniques

2.6.2.1 Inverted File

In the past many information retrieval systems used a direct file organization that stored individual files in some predetermined order. When a search request was submitted, it involved searching the full text of each document in the collection. This approach can be an excellent choice when performing batch processing of queries since only one entire scan of the collection would be required. However, there is a major drawback to the direct file approach. In most information retrieval systems, queries are submitted one at a time, so an entire scan of the documents for each query can be quite slow.

When queries are submitted one at a time, the use of some indexing structure to represent the document collection is more appropriate. There are many different methods to create the index. Two of the more well-known methods are inverted files and signature files. However, [Zobel, 98] states that inverted files are superior to signature files because of less space requirements, providing faster query evaluation, and providing greater functionality. Inverted files are a file organization where the keywords are indexed with a reference to each document the keyword appears in, as seen in Figure 7.

As seen in Figure 7, the word “inverted” appears in three documents within the collection (Documents: 1, 3, and 6). The document key is not the only information that can be stored with each index term. As we will see later, storing the number of times the word appears in the document will also be useful. The benefit of the inverted file is that it is only necessary to look at documents that include the individual index terms that are contained in the query, resulting in a quicker query response time. However, in the worst-case scenario, indexing every word in the collection can result in an index as large as the collection itself. Two approaches are now discussed that can help reduce the index size.

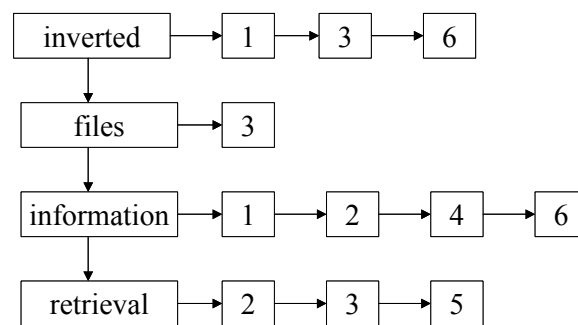


Figure 7. Conceptual View of an Inverted File

2.6.2.2 Stop Words

Stop words are words that appear in many of the documents within the collection. Words like 'the', 'and', 'a' etc. will likely appear in every document. Thus, the ability of these words to discriminate between documents is impossible. Therefore, removal of these words from the index can reduce the index size considerably. In addition to articles and prepositions, stop words can be generated from the index terms within the index itself. So cutoff point can be established (i.e. a word appears in more than two-thirds of the documents) to ignore the word, should it be submitted in a query. However, removal from the index can create problems in the future. For instance, more documents are added to the collection, not containing the current generated stop words. If enough documents are added, then the stop words become more relevant and are no longer stop words and should be included in the query.

2.6.2.3 Stemming

Stemming is another approach to reduce the size of the index. It is the process of stripping a word of prefixes and suffixes to get to its root. Only the root is stored in the index along with the document keys of the all the variations of the words in the collection. For example, the words, psychiatrist, psychiatry, psychiatric, psychology, and many other terms reduce to form PSYCH*. In addition to reducing the index size, stemming is also used to improve recall by generalizing over word variants [Riloff, 95]. However, experiments in the use of stemming by [Harman, 91] and [Krovetz, 93] have produced mixed results. There is a number of ways to strip the prefixes and suffixes

from words. One approach is to remove suffixes is to remove only pre-selected ones determined before indexing; the other is to remove a fixed number of characters from the end of the word. The goal is to remove suffixes to form reasonable recognizable terms. For example, “ing” would be removed from the word “indexing,” but not from the word “king.” Removing prefixes can be reduced to suffix truncation, if the words are indexed in the correct manner. One way to accomplish this is to index the words backwards. For example, the word “antisymmetry” would be entered as “yrtemmysitna.” However, to index words that contain both prefixes and suffixes, the words must be indexed forwards and backwards [Salton, 89].

2.6.2.4 Thesauri

One of the main issues in information retrieval is the language variability between authors and users [Salton, 71]. This problem occurs when there is a word-mismatch between the words contained in the document collection and the user’s query. This problem tends to be less pronounced as the query gets longer. However, in most applications queries tend to be only a few words. For example, [Croft, 95] states that queries on the World-Wide Web tend to be only two words on average. The most common approach to alleviate this problem is through the use of a thesaurus. A thesaurus or “synonym dictionary” can replace index terms with words of the same meaning. Many different methods of thesaurus creation have been tried over the last three decades and can be grouped into three general categories [Mandala, 99]. Thesauri are either generated by hand or automatically through a co-occurrence based method or a head-

modifier method. Thesaurus use helps alleviate the word-mismatch by expanding the user's query by adding words that may mean the same to the query. However, the increase in retrieval performance can depend on the thesaurus and IR system pairing or the thesaurus alone. Constructing a thesaurus in a given subject area can be accomplished automatically, but is best accomplished manually through a group of experts [Salton, 89]. In addition to storing synonyms, a hierarchical thesaurus on concepts can be constructed to broaden or narrow a search request. An example of a hierarchical thesaurus can be seen in Figure 8 below.

[Salton, 71] states that broadening through parents or narrowing through children, can improve retrieval performance for certain recall levels, however, a standard thesaurus alone produces better results. This is because a query expansion using a hierarchy can crystallize the meaning of a poorly stated query and the change in the direction caused by incorporating a hierarchy can be too violent.

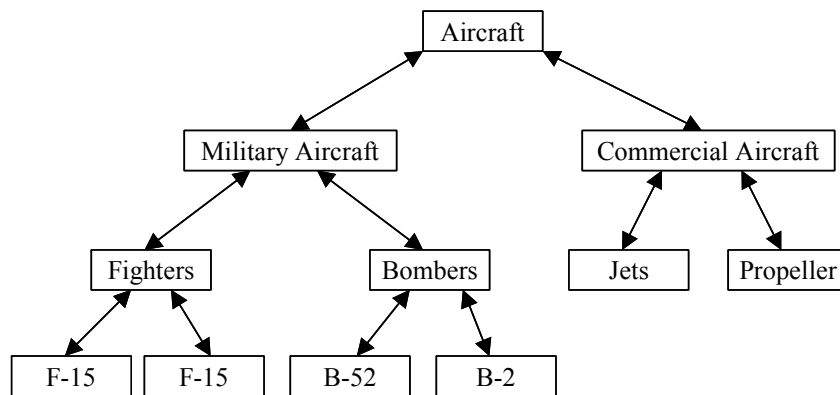


Figure 8. A Partial Hierarchy of Aircraft Words and Categories

2.7 B-Trees

The index of the document collection must be stored in persistent memory for use at later times. B-Trees are balanced search tree data structures that minimize the time of disk I/O operations [Cormen, 90]. The logical view of a B-Tree can be seen below in Figure 9. This figure shows that reaching any letter in the alphabet will never take longer than two reads. Whereas, in a strict binary tree reaching some letters could take as many as five reads, thus causing the program using the tree to wait on I/O.

The size of the inverted file can grow very large, however, as more documents are added to the collection the number of index terms in the file grows slowly. This is due to Heap's Law where it states that a vocabulary of size n words is of size $V = Kn^\beta = O(n^\beta)$. Normally, K is between ten and one hundred and β is between zero and one. Values of K and β depend on the size of the text, however, an experiment on the TREC-2 collections demonstrated that the vocabulary grows in proportion to the text size, close to its square root.

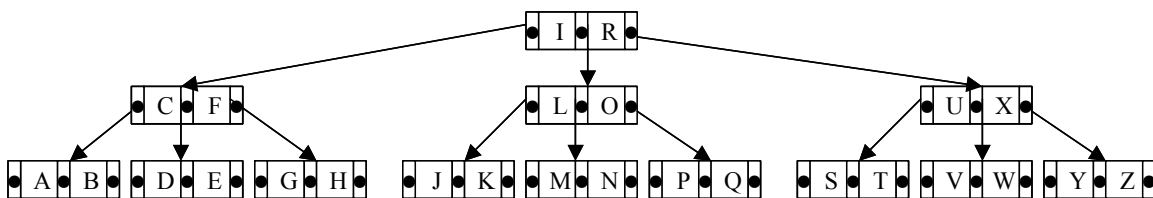


Figure 9. B-Tree Logical View

2.8 Summary

This chapter presented a review of the various technologies that form the cornerstone of this research. First the overall structure of the VDL concept was

examined, followed by definition of metadata and its role in this research. Next, an introduction to relational database theory and a brief synopsis of the capabilities and drawbacks of SQL is presented. A brief discussion follows of heterogeneous databases and the hurdles to overcome to retrieve information from multiple heterogeneous databases. An introduction to information retrieval, its few retrieval models, and some simple techniques are discussed next. This chapter concludes with an overview of the composite pattern and a review of B+ Trees. In the next chapter, the methodology used to incorporate these technologies to retrieve relevant images from the VDL On-Line Library is presented.

III. Methodology

3.1 Introduction

This chapter defines the methodology used to develop an image retrieval system capable of providing users an effective means to query and retrieve images in order of relevance from metadata contained with multiple heterogeneous relational databases.

Currently, there is no mechanism within the VDL environment to map between the heterogeneous databases. This includes the database schema and data values within the database. The approach taken is to incorporate the use of a hierarchical thesaurus, where a global schema and data values can be developed to perform query expansion to include words that appear across more documents within the collection. This research develops two tools, that when working together can develop the hierarchical thesaurus and use the thesaurus to retrieve relevant documents from the databases.

This chapter begins with an overview of the tools used to perform the analysis, design, and implementation of the Schema Integration Tool and the Query Tool. This is followed with a discussion of the general approach used in the Schema Integration Tool. Here, the concepts of document representation, database normalization, database parsing and indexing are discussed. Following this is a discussion of the methodology used to develop the Query Tool with regards to filtering and relevant document retrieval. The chapter concludes with the discussion of the design of the Schema Integration Tool followed by a similar discussion of the design of the Query Tool.

3.2 Development Tools

The development of the Schema Integration Tool and the Query Tool utilizes a call and return program architecture using the Java programming language. This section describes the capabilities and advantages of each.

3.2.1 Call and Return Architecture

The Call and Return programming architecture is a type of structural programming model that provides the programmer a structure that is easy to modify and scale [Pressman, 01]. With this architecture there is a “main” program method from which additional methods may be called. These additional methods may in turn call other methods to accomplish their goal. Eventually, the program control is returned to this main function so that additional method calls may be made.

3.2.2 Programming Language

The Java programming language is used as the implementation language for the Query and Retrieval Tool and the Schema Integration Tool. Java was developed by Sun Microsystems to be an easy to understand programming language that provides many benefits [Sun Microsystems]:

- **Platform Independence:** Any Java program can be compiled and executed on any Java Virtual Machine (JVM) regardless of the underlying hardware or operating system.
- **Exception Handling:** Java supports throwing and catching exceptions at the hardware and program level.

- **JDBC Interfacing:** Java provides the JDBC database interface to make simple database tasks, such as establishing a connection, querying, and retrieving results very easy.
- **JFC/Swing:** Java provides an easy manner in which to build Graphical User Interfaces (GUI), through its Java Foundation Classes (JFC). Within the JFC, developers can incorporate everything from buttons to split panes to tables into their GUIs.

With these features, Java provides a powerful set of programming tools to facilitate the implementation of GUIs to access databases through the object-oriented programming paradigm. This is all accomplished while having the capability to be compiled and executed on any system containing a JVM.

At the time of this research, Sun Microsystems' most recent version of the Java language is the Java Development Kit 1.3 (JDK 1.3) and is the one utilized in this research for the implementation of the Query and Retrieval Tool and the Hierarchical Thesaurus Creation and Indexing Tool.

To facilitate the development process, JBuilder was the environment used to implement the design of tools in this research. JBuilder is a Java development environment produced by the Borland Software Corporation that provides a point and click GUI building environment and excellent code generation facilities.

3.3 Hierarchical Thesaurus and Indexing Tool

3.3.1 General Approach

This section discusses the general approach used to develop the Schema Integration Tool used in this research. First the approach used to represent documents is

presented, followed by the approach used to normalize data values and column headings. Finally, this section concludes with a discussion of the Schema Integration Tool functions.

3.3.2 Document Representation

To retrieve relevant images within a relational database using text-based information retrieval techniques, the Schema Integration Tool uses a unique document definition strategy. Since each record in a relational database is assumed to be unique, each record is treated as an individual document. Also, the entries of the record describe the attributes of a single image, much like words of a textual document describe the document itself.

3.3.3 Hierarchical Thesaurus

Differing schemas and data values are the major hurdles to overcome to retrieve items from heterogeneous databases. The Schema Integration Tool uses a hierarchical thesaurus to alleviate the problems caused by this type of environment. Using a hierarchical thesaurus not only allows users to search for documents without knowing the exact vocabulary used, but also allows them to search for higher-level concepts that may not appear textually in a document.

To provide the VDL administrator a basis to create higher-level concepts the idea of a category is introduced. A category is an object that can contain words, and other categories. It has a one-to-many relationship with words and a many-to-many relationship with other categories. This means that each category has the potential to

contain many words, however, words may only be assigned to one category. These relationships can be seen in Figure 10. The many-to-many relationship means that a category may be contained within other multiple categories, thus, creating a hierarchy of more general concepts. Figure 10 shows that F_16 and F16 are individual words that have the same meaning. However, a category named F-16 has been created to group these words together as synonyms. At this level, the category follows the same principles as a thesaurus. If a user entered a word belonging to a category, the query would be expanded to include the other words that were maintained in the same category. Without this approach, relevant documents would go unnoticed due to the differences in vocabulary between the query and the words in the actual document. However, the hierarchical thesaurus takes the concept of a standard thesaurus one step further. Also seen in Figure 10, the F-16 category has been assigned to multiple higher-level categories. These categories serve as higher-level concepts of the individual words. This allows the user to define a more general concept during query time, even if the concept words are not found in the document itself.

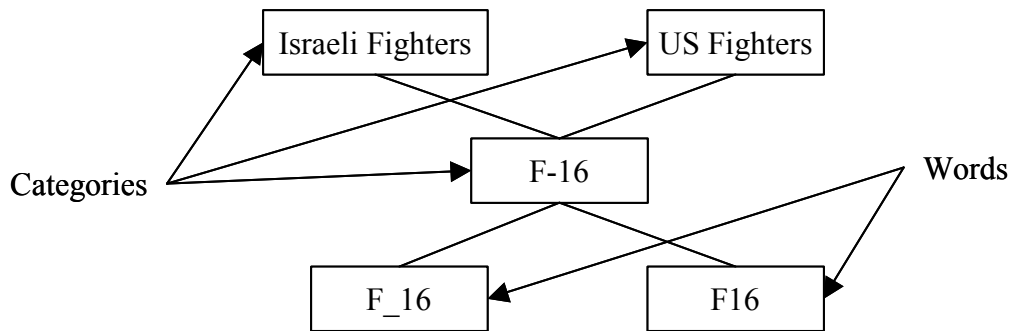


Figure 10. Hierarchical Thesaurus Example

By selecting concepts higher in hierarchy of the thesaurus, users may query for more general concepts such as “tanks” without the word actually appearing in the document. As the user moves down the hierarchy, their queries become more and more specific until they reach a specific word.

As discussed in Chapter 2, stemming is another method to retrieve possibly more relevant documents and can be accomplished using the hierarchical thesaurus. The thesaurus may contain a category based on the root of a word, such as “like.” Variations of this word may include “unlike” and “likely.” Words with prefixes and suffixes may be assigned as synonyms to one another under the root word category. Users may use these methods to search documents without regard to the verb tense (past, present, future) the document was written in. However, this stemming approach does not save any index space, unlike normal stemming where only the root word is stored and stemming algorithms are used to recreate words based on the root.

To be able to assign words as synonyms through the use of categories, each document must be parsed to determine what words are contained within it. The approach behind parsing a document used in this research is discussed in the next section.

3.3.4 Database Parsing

As discussed previously, a document is considered to be a single record in a relational database consisting of all columns. Parsing a database consist of finding all data values, excluding numerical values, and presenting them to the VDL administrator for possible inclusion to a category.

However, not all words may be included. Some stop words (i.e. “a”, “and”, “the”, etc.) may not be included at the discretion of the VDL administrator. Note however, that the use of the stereotypical stop words are not recommended due to the type of object the documents in this research are describing. In imagery descriptions, the use of prepositions as stop words can significantly reduce the level of recall for a system [Riloff, 95]. This is due to their precise descriptive value. For example, if a user entered a query to see images with all “tanks on a bridge.” If the standard practice of removing prepositions is followed, then the query terms are reduced to “tank” and “bridge.” This approach treats the images of “tanks on a bridge” and “tanks next to a bridge” as equivalent. Obviously, this is not what the user intended. If the preposition “of” were indexed then the documents containing the words “tank”, “bridge” and “of” would be scored higher since “of” would add extra weight to the documents than those without it. For this reason, prepositions are not considered stop words and should be indexed. Word weights are an integral part of any textual based information retrieval system. The approach used to set individual word weights used in this research is discussed in the next section.

3.3.5 Indexing

The idea behind indexing is to track which words appear in which documents along with their frequency within individual documents and throughout the collection. The indexing approach used in this research is accomplished through two-steps. First, each document is indexed similar to an inverted file. The next step is to index the column

headings as to which words appear in their column. These two approaches are discussed in the following two sections.

3.3.5.1 Document Indexing

Document indexing follows the same general principles as parsing the database to find individual words, with two exceptions. First, when the initial set of words are presented to the VDL administrator; multiple instances of the same words are ignored. However, when a document is indexed, as words appear more often throughout the collection, their occurrence list is updated as to which documents they are contained in and their frequency of occurrence. This information is used to set the *idf* of individual words that are used in the two information retrieval models, Vector and Extended Boolean, as discussed in Chapter 2. However, using a thesaurus creates problems when calculating the *idf* of words and the maximum word frequency in a document. Words having the same meaning, through the assignment to the same category should have the same *idf* factor since they are treated as being identical. Recall that:

$$idf_i = \log \frac{N}{n_i}$$

Where, N = total number of docs in the collection and n_i = number of docs index term k_i appears.

Setting the *idf* of words within the same category requires an additional pass through the list of words to determine which words are assigned to categories. If a word

is assigned to a category, then the *idf* of the word is the summation of the number of documents it and its synonyms appear in.

3.3.5.2 Column Headings

The idea of required criteria that a document must contain before it can be deemed even partially relevant is also important. This is similar to the idea of SQL and the Boolean model in information retrieval. With this approach only documents that meet all criteria are scored for relevance. This is accomplished through indexing the individual columns of a database. When a word or number is found by the parser then the occurrence list of the column to which it is assigned is updated with that value and the documents in which it appears. In the Query Tool, the user may select criteria that has been indexed in this manner as a way to filter unwanted documents that do not meet the criteria.

3.4 Query Tool

3.4.1 General Approach

This section discusses the general approach used to develop Query Tool used in this research. The approach used to retrieve relevant documents is presented first, followed by the approach used to rank relevant documents.

3.4.2 Relevant Document Retrieval

Relevant document retrieval is the process of retrieving documents the user may be interested in. The manner in which this research determines relevant documents is

accomplished in two steps. With the first step, the user may reduce the number of documents retrieved by requiring that certain criteria be met in the description of the image, regardless of its location within the database. This approach is simply an implementation of filtering similar to the Boolean information retrieval model. In the Boolean model, only documents meeting all criteria are returned as relevant documents. However, as with Boolean model, no ranking of the documents is provided, and therefore all returned documents from this filtering are equally relevant. If a user enters only filtering criteria and no query terms then the user has essentially issued a Boolean query with the AND operator between each criteria, and will cause the above situation to occur. As one would expect, the collection of relevant documents becomes smaller as more mandatory criteria is placed on the documents that must be met.

If the user has entered a series of keywords as a query, then only the collection of documents meeting the criteria are ranked by relevance. If no query is provided then all documents are considered relevant that meet the filtering criteria are relevant.

If a user has entered a query, then the next step involves the parsing the user's query to determine keywords by which to evaluate the documents. At this point, the query is expanded through the use of the hierarchical thesaurus. If any of the query words are members of categories, then the other words within the category are included in the query. Also, if a category is entered as a query word, then all words below it are also included, regardless of their location in the lower sub-tree. Each of the remainder of the relevant documents must contain at least one of the keywords in the query to remain. If no keyword in the query is found in the document, then it is removed from the relevant

document set. This process of elimination of documents that have no relevance to either the user’s criteria or the keywords in the query reduces the time to return the relevant documents to the user.

Once the process of determining possible relevant documents is completed the documents are individually ranked by the methods presented in the next section.

3.4.3 Relevant Document Ranking

Once the set of relevant documents has been determined and a query has been issued, the documents are ranked according to one of the two algorithms discussed in Chapter 2. Both the Vector and Extended Boolean model provide a natural ranking of the documents according to their relevance to the query and document criteria.

3.5 Design Architecture

In the following two sections the designs of Schema Integration Tool and the Query Tool are presented. Figure 11 provides an overview of how these tools are used together to accomplish the goal of retrieving relevant documents. As the figure shows,

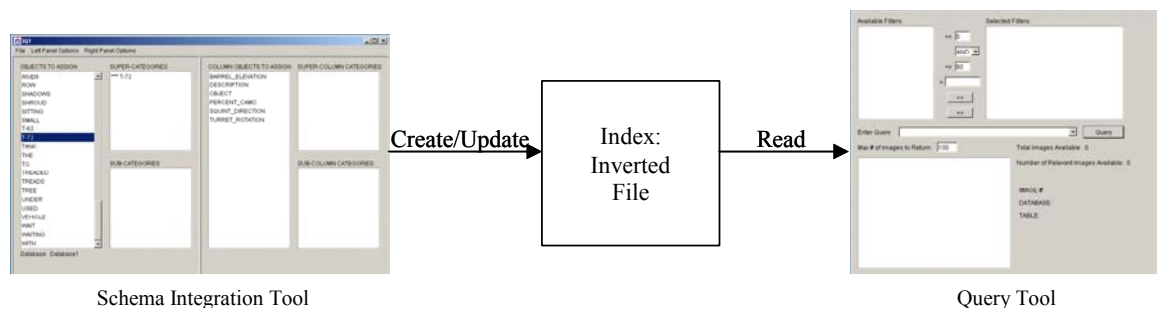


Figure 11. Pipeline Architecture Between the Two Tools

The Schema Integration Tool produces and updates the inverted file based on the words found in the database documents and the assignment of words to categories and the hierarchy of categories. The Query Tool uses the inverted file to provide the user a way to retrieve documents ranked by their relevance to the submitted criteria and query. As one would expect, the performance of the Query Tool can only be as good as the inverted file it is based on. In the following sections, the designs of these two tools are covered in detail.

3.5.1 Schema Integration Tool

The architecture of the Schema Integration Tool is written using a functional paradigm comprised of thirty-two method calls as seen in Figure 12. The call structure can be seen in Figure 13. The major functional pieces that are cutout are explained in the following sections.

1	jbInit()	17	RemoveColumnCategory()
2	QuitProgram()	18	DisplayAddSuperCategoryPopup()
3	ShowConfirm()	19	DisplayAddSuperColumnCategoryPopup()
4	InitialConnection()	20	DisplayRemoveSuperCategoryPopup()
5	ChangeConnection()	21	DisplayRemoveSuperColumnCategoryPopup()
6	ReadCategoryInformation()	22	AddSuperCategory()
7	ReadColumnCategoryInformation()	23	AddSuperColumnCategory()
8	ResetObjectsToAssignList()	24	RecurseDown()
9	ResetColumnObjectsToAssignList()	25	RemoveSuperCategory()
10	ResetSubCategoriesList()	26	RemoveSuperColumnCategory()
11	ResetSubColumnCategoriesList()	27	GetObjectsToAssign()
12	ResetSuperCategoriesList()	28	IndexDatabase()
13	ResetSuperColumnCategoriesList()	29	SetSynonyms()
14	AddCategory()	30	SetWordIDF()
15	AddColumnCategory()	31	SetCategoryInfo()
16	RemoveCategory()	32	RecurseFunction()

Figure 12. Available Method Calls of the Schema Integration Tool

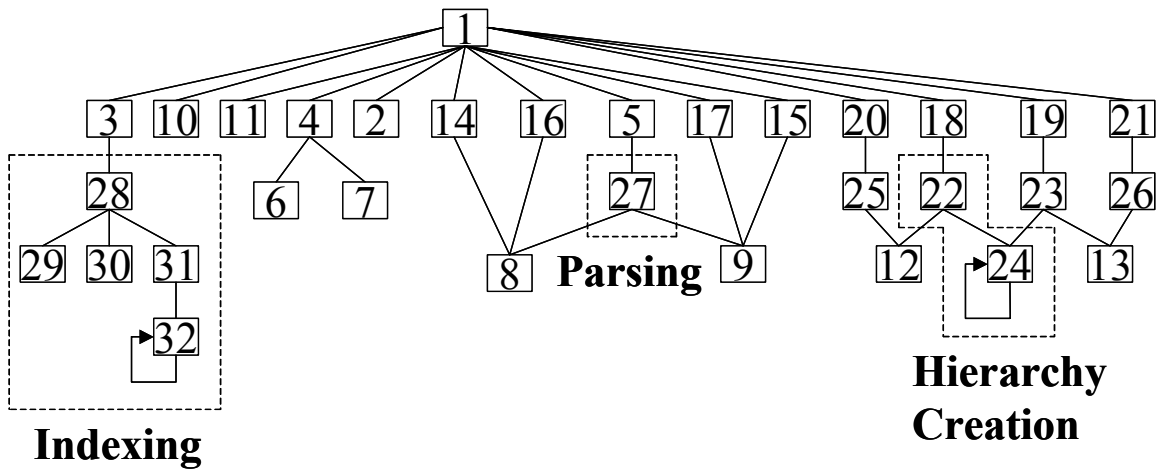


Figure 13. Schema Integration Tool Method Call Chart

3.5.1.1 Parsing

Once a connection to a database is established, the *GetObjectsToAssign()* method is responsible for parsing the current database. The database parsing consists of two main functions. First the *GetObjectsToAssign()* method retrieves the individual column headings of the relational database. It is also responsible for extracting each unique word (i.e. all non-numeric and non-null values) and presenting them to the user for possible assignment to categories. Once the column headings and the distinct words have been determined, then the *GetObjectsToAssign()* method resets the graphical user interface.

3.5.1.2 Hierarchy Creation

The hierarchical thesaurus consists of two distinct hierarchies, one for column headings and one for data values within a database. The manner in which these two distinct hierarchies are created is accomplished in the same manner. Assigning a category to a word is relatively easy: if a word has not been assigned a category, then update the word with the category it is assigned to and add the word to the list of children

of the category. If a word has been assigned to a category previously, then reassign it to the new category, updating the word, the new category, and the old category about the change. Additionally, adding and removing categories are relatively easy. When removing a category, only its children and parents within the hierarchy must be updated to reflect the removal of the category. However, it is the assignment of one category as a parent to another category that can be somewhat tricky. When adding a parent category certain rules must apply. The most important is that the new addition may not create a cycle in the graph, meaning the category to add as a parent may not already be a descendent of the child category. In Figure 14, it can be seen that the category “Israeli” could not be assigned as a parent to the “Fighters” category because this would create a cyclical graph. However, also seen in Figure 14, a category may also have a sibling as its parent in that the category F-16 may be a child under “Israeli”, but is may also be a child directly under the “Fighters” category.

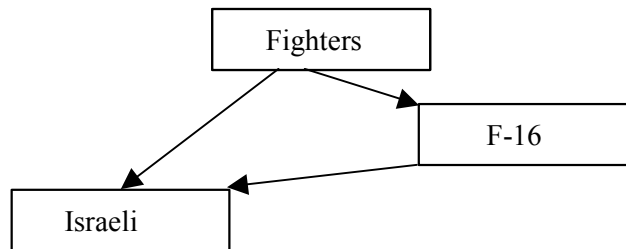


Figure 14. Preventing Category Cyclical Graph Example

3.5.1.3 Indexing

The *IndexDatabase()* method is responsible for re-parsing the database and setting word weights based on the occurrence in the document collection. This method’s primary responsibility is to parse and update each word’s and each column’s occurrence

list as to which document it appears in and frequency of occurrence. It uses three additional utility functions to inform each word of its synonyms, its *idf* weight, and updating the categories with all relevant words and documents that are located as children below it in the hierarchy. Once the database has been parsed, then the words are iterated over to update their synonym lists by looking at their assigned category and the additional words assigned to the same category. Computing the *idf*, as discussed earlier, is the process of determining the number of distinct documents a word or one of its synonyms is located within. Since each category may have both categories and words as children, determining the relevant words and documents in its sub-tree requires the use of a recursive function to do a depth-first search to retrieve this information. This approach requires the storage of repetitive data, however, it is used to increase the speed of the query tool by not having to perform this operation at query time.

It has been noted that not all columns need to be searched. Therefore, in an effort to increase indexing speed, only columns selected by the VDL administrator are indexed. This approach not only increases speed, it also reduces the size of the disk-based B-Tree, therefore saving hard disk space. In addition, a reduction in the B-Tree size decreases lookups to retrieve individual words, thus increasing the speed of the Query Tool. A discussion of the architecture of the Query Tool is provided in the next section.

3.5.2 Query Tool

The architecture of the Query Tool is also written using a functional paradigm comprised of thirteen method calls as seen in Figure 15. The call structure for the Query

Tool can be seen in Figure 15. The *BasicSearch()* method is the primary method within the Query Tool and its responsibilities and additional method calls are discussed in the following sections.

1	main()	8	VectorModel()
2	DisplayCategories()	9	ExtendedBooleanModel()
3	DisplayDocInfo()	10	RecurseEquals()
4	AddFilter()	11	RecurseBoth()
5	RemoveFilter()	12	RecurseGreater()
6	SetBasicQueryInterface()	13	RecurseLesser()
7	BasicSearch()		

Figure 15. Available Method Calls of the Query Tool

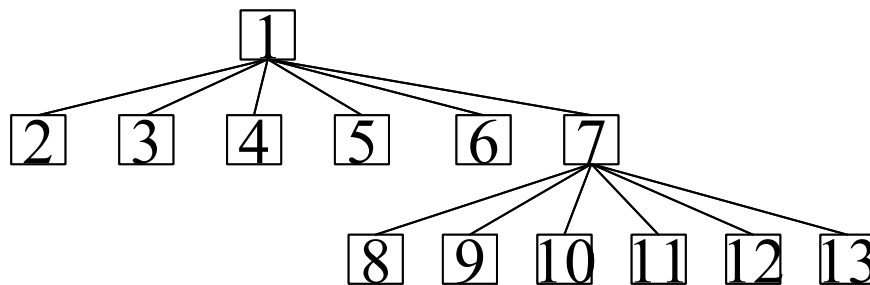


Figure 16. Query Tool Method Call Chart

3.5.2.1 Filtering

The first half of the *BasicSearch()* method involves filtering of documents that do not meet the specified criteria by the user. No individual column headings are displayed to the user; only “Column Categories” are displayed. Therefore, if a column heading is to be searchable, then it must be assigned to a higher-level category. The *BasicSearch()* is responsible for parsing the user’s criteria into smaller pieces of information. This information includes which Column Category the user has requested, and the specified ranges to include. The filtering mechanism provides the user the ability to search for a

specific range of inclusive numbers or an individual word that must be present within the column heading. However, multiple word searching within the same column heading is not currently implemented.

The filtering approach was selected over a pure information retrieval approach. In a pure information retrieval approach, an algorithm would be developed to determine a weighting factor for the criteria based on the distance from the user's original request, thus treating the criteria similar to a query term. This primarily deals with the instance of where the user has selected criteria that must be equivalent to some number. This approach could provide smaller weights as the values of the characteristics of the documents move farther from the user defined number. However, it is generally understood that when a user specifies that a document must meet some criteria, then any document that does not meet any one of the specified criteria is excluded.

3.5.2.2 Relevant Document Ranking

Once the document collection has been filtered and only documents that meet the strict criteria from the user remain, the process of ranking these documents according to their relevance to the query can begin. This ranking is accomplished through the selection of two classic information retrieval models (Vector and Extended Boolean). These models provide a natural ordering of the documents in ascending order according to their relevance. The implementation of these information retrieval models is discussed in the next chapter.

3.6 Summary

This chapter begins with an overview of the design architecture used in the implementation of the Schema Integration Tool and the Query Tool and discusses the advantages of using the Java programming language in this research. The Schema Integration Tool has three general functional requirements of document representation, and parsing and indexing the individual databases. Within the Schema Integration Tool the VDL administrator may also develop high-level concepts of words that may be contained in the documents and assign synonyms to standardize the data values found in the database. This will allow the users to search for concepts and words that may not be contained directly in the text of the documents.

The functional requirements of the Query Tool consist of permitting the user to filter documents according to some specified criteria before beginning the ranking process based on query terms. In addition, the Query Tool is responsible for retrieving the documents that contain at least one query term from the subset of documents meeting all the user's criteria. This subset of documents is ranked via the Vector or Extended Boolean model according to relevance. The design of the Schema Integration Tool and Query Tool are based on a Call and Return Architecture permitting easy modifications to the program as needed. In Chapter 4 both applications are implemented, tested, and the results interpreted.

IV. Implementation and Results

4.1 Introduction

This chapter discusses the functionality of the Hierarchical Thesaurus & Indexing Tool and the Query Tool as implemented in this research. It begins with a discussion of some important design issues that were encountered during the design and implementation stages. Next, the two major functional areas that make up the Hierarchical Thesaurus and Indexing Tool are presented. These two areas provide all the functionality to provide hierarchy creation and document indexing. This is followed by a discussion of the implementation of the Query Tool. Finally, a brief discussion of the evaluation criteria is presented followed by the test cases and results.

4.2 Design Issues

During implementation, two key issues came to light and a brief discussion of each is presented here. These areas include the speed and the amount of memory used in creating and maintaining the indexes and hierarchies, in addition to relevant document retrieval.

4.2.1 Memory

The amount of space taken up by the document index and categories hierarchy is of great concern when dealing with the large volume of documents in the VDL environment. When dealing with a small number of documents, the index and hierarchy can be held in main memory. This approach would greatly increase the speed by which

the hierarchies can be maintained and relevant documents could be retrieved. However, as the number of documents in the collection continues to grow, it becomes increasingly difficult to maintain these structures in memory and therefore, this approach was deemed not feasible and was not pursued. Since the index cannot be held in main memory, a persistent storage mechanism must be implemented to maintain a large index and hierarchy while allowing fast access to data when needed. In addition to eliminating the requirement to rebuild the index each time the tool is executed, persistent storage provides a medium by which the document collection's growth is not bounded by the limit of main memory, but only by the amount of available disk space. Therefore, the B-Tree data structure is the data structure of choice in this research to maintain the long-term storage of the indexes and hierarchies. The B-Tree that was used for this research stores nodes on individual pages on disk. If a node is larger than the page size, then it is stored across multiple pages.

4.2.2 Speed

Speed was one of the most critical factors when developing the tools in this research, specifically for the Query Tool. In the Hierarchical Thesaurus and Indexing Tool, the speed at which documents are indexed is not crucial to the success of this research, but should be considered. On the other hand, the speed at which the Query Tool can return relevant documents should not grow proportionally to the growth of an increased document collection size. Results of whether this was accomplished will be seen later in the chapter.

4.3 Hierarchical Thesaurus and Indexing Tool

The design of the Hierarchical Thesaurus and Indexing tool was developed with three major functions in mind. These functions and the manner in which there are implemented in this research are discussed in detail in the following sections.

4.3.1 Parsing Documents

Before a user can create and assign synonyms to individual words in a document through categories, each document must be parsed to determine the unique set of words that make up the database in the collection. For this, Java's JDBC is used to connect to the desired database and records are read one at a time and added to a data structure for presentation to the user. If column values are found to contain multiple words or phrases, these are broken down to their atomic words via a string tokenizer and also inserted into the data structure. Only words are considered, if a data value is determined to be a number then it is ignored. In addition to parsing documents, the parsing function is also responsible for determining the values of the column headings and also presenting them to the user. Once the parser has determined the set of unique words and the database column headings, each value is sent to their respective B-Trees on disk. There are a total of five B-Trees, one for the regular words, one for the column headings, one for word categories, one for column categories, and also one for a unique database and table combination identifier to be used during the indexing process. The end result of the parsing function is two lists of words that may now be assigned to different categories.

4.3.2 Hierarchy Thesaurus Generation

Categories may either be created before or after a database has been parsed. However, assignment of words to categories may only be accomplished after the database has been parsed. If a word has been assigned to a category from an earlier database, if a new database is loaded and parsed and the word appears again, there is no need to reassign the word to the same category. Although words may only be assigned to one category, categories may be assigned to multiple categories. Once the VDL administrator has determined the thesaurus is complete, the indexing process may begin. If a database has been indexed prior to any changes made to the thesaurus hierarchy, there is no need to index the database again. The indexing of the current database will handle any additional category assignments or re-weighting of terms.

4.3.3 Indexing

Indexing the database initially begins with the parsing of documents similar to the method discussed above, with the exception of ignoring duplicate terms and numerical values. As stated earlier, each database and table combination within the document collection has been assigned a unique number. This number, along with the document key within the database, is used to create a unique document identifier to determine its location if it is retrieved as a relevant document.

Two types of indexing take place requiring only one run through the documents in the database. The first type of indexing that occurs is column indexing. Column indexing is used by the Query Tool to simulate the actions of the Boolean information

retrieval model. As values, including numerical, are parsed, they are inserted under the appropriate column word along with the document identifier in which they appear. An example of this architecture can be seen in Figure 17.

In addition, the inverted file must also be updated to include the new copy of the word with the document identifier in which it was found for later use in the IR models discussed below. The individual word architecture is a simplified version of the column word architecture as the column heading is ignored and the word becomes the word identifier and not just a value. However, numerical values are ignored. An example of this simplified architecture can be seen in Figure 18.

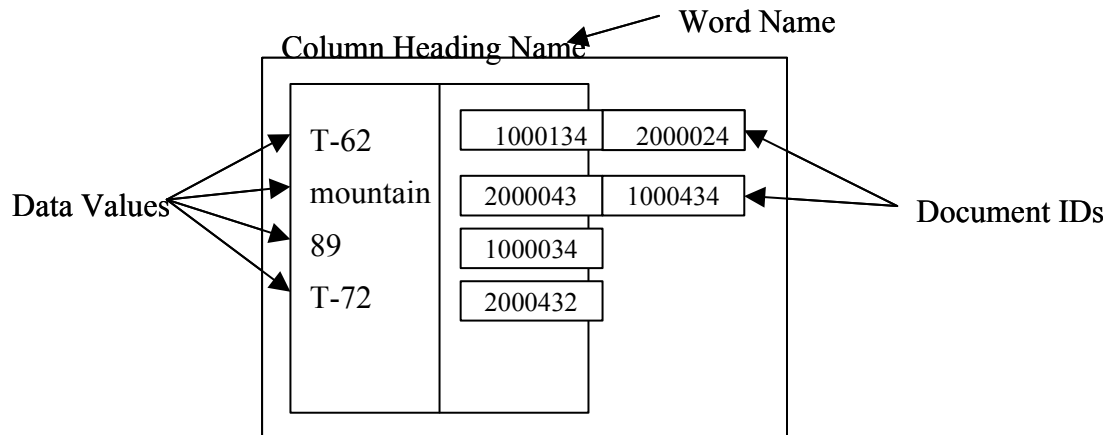


Figure 17. Column Word Architecture

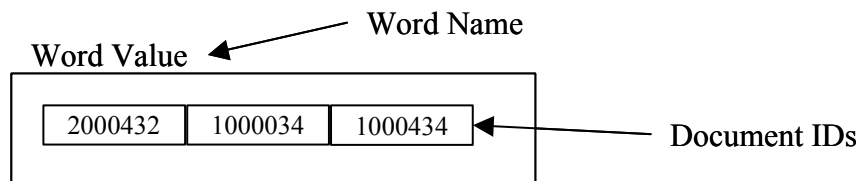


Figure 18. Word Architecture

As one might expect, the occurrence list of documents in which the word and column word values appear can grow very large as they appear in more documents. A word with an occurrence list of any substantial length can require a significant amount of time to load into memory. In addition, the word then must be written back out to the B-Tree on disk. This amount of disk I/O can significantly slow down the indexing process. Therefore, the idea of an external cache was implemented to limit the number of reads and writes to disk. The VDL administrator determines the cache size based on the amount of memory available on the computer on which the indexing tool resides. This is accomplished by specifying the maximum number of documents to be indexed and held in memory at any moment in time. A typical computer can handle at least 10,000 documents in memory at a time. Once a word is parsed, the cache is checked to see if it has already been loaded from the B-Tree on disk. If the word is found in the cache, then the word's occurrence list is updated and saved back to the cache. However, if the word is not presently in the cache, then it must be loaded from disk, updated, and saved back to the cache. Once the number of documents has been indexed that was specified by the VDL administrator, the entire cache is written to disk and cleared. It is written out to disk by iterating over each word in the cache and loading the same word from the B-Tree. The difference in the occurrence lists are updated from the word that was loaded from the B-Tree and then written back out to disk. This significantly reduces the frequency of writes to and from disk. The hash map data structure was selected due to its $O(1)$ access time to determine whether a word is present in the cache and its $O(1)$ retrieval time to

load a word from cache. Once the inverted files of words and column words have been completed, the assignment of word weights takes place next.

Only the number of documents the word appears in versus the size of the collection determines the word's weights when not using a thesaurus. However, when using a thesaurus, all words that are determined to be synonyms must contain the same weights. Therefore, each word in the inverted file is expanded to take into account its synonyms and the documents in which they appear. The word's *idf* is calculated using this approach, however, the occurrence lists of the words are not changed to reflect the additional documents its synonyms may be present in.

In addition, all categories in the hierarchy also contain a listing of all words that are contained under its sub-tree. The category also contains all documents that are in the occurrence lists of these words. This approach saves time during the query process by performing the necessary calculations during indexing and not during the query process.

4.4 Query Tool

The design of the Query Tool is developed with two major functions in mind. These functions and the manner in which they are implemented in this research are discussed in detail in the following sections.

4.4.1 Relevant Image Filtering

Relevant image filtering involves the use of the column word index that was created to mimic the attributes of the Boolean IR Model. Users may select columns that

are of interest to them. However, only column headings that have been assigned to categories are available. In essence, the user selects a category that may contain multiple column headings or other categories to filter through documents. Documents that do not meet all the users' criteria are removed from the possible relevant document set.

4.4.2 Relevant Image Ranking

Once the filtering of documents has taken place and all documents that do not meet the user's criteria have been removed, the remaining documents can be scored and ranked according to their relevance. However, when using the thesaurus, the user may expand the query by selecting a category within the hierarchy. All words in the selected categories' sub-tree are added to the users query. The documents that contain at least one of the words found in the query are kept; all others are removed since their relevance scores will be zero. If the user does not enter a query, then documents that meet the user's specified filtering criteria are returned with no ranking. Now there are only documents that have at least one query term and have met the user's criteria, the scoring and ranking of these documents may be accomplished through the use of one of the following algorithms.

4.4.2.1 Vector Model

The Vector model was implemented as presented in Chapter 2, with a few minor adjustments. The Vector model's formulas are shown below:

$$f_{i,j} = \frac{freq_{i,j}}{\max_i freq_{i,j}}$$

Where, $freq_{i,j}$ = raw frequency of term k_i in doc_j and $\max_i freq_{i,j}$ is the maximum freq. of all terms in doc_j , and

$$idf_i = \log \frac{N}{n_i}$$

Where, N = total number of docs in the collection and n_i = number of docs index term k_i appears

The index term weights are calculated by multiplying the results of the two formulas together as seen here:

$$W_{i,j} = f_{i,j} \bullet idf_i$$

Where as the query weights were calculated using the following formula:

$$W_{iq} = \left(0.5 + \frac{0.5 * freq_{i,q}}{\max_l freq_{i,q}} \right) * \log \frac{N}{n_i}$$

One of the changes involved the simplification of the query weights. The assumption was made that the user would only enter the query word once, therefore reducing the query weights to the following:

$$W_{iq} = \left(0.5 + \frac{0.5 * 1}{1} \right) * \log \frac{N}{n_i} = \left(0.5 + \frac{0.5}{1} \right) * \log \frac{N}{n_i} = \log \frac{N}{n_i}$$

In addition, the formula for the term weights were interpreted slightly differently than in Chapter 2. This alternative approach is to only consider the maximum frequency in a document from the query terms and not the entire document itself. The previous

approach tended to give an advantage in scoring to shorter documents since longer documents may contain more frequent words that were not contained in the user's query. When only considering the terms in the query, the size of the document will have less of an influence on its score.

4.4.2.2 Extended Boolean Model

The Extended Boolean model as implemented in this research uses the P-norm value as discussed in Chapter 2. The user has the option to select the value of p from one to five to simulate the Vector model or to use a more Boolean approach. In addition, the AND operation was used exclusively over the OR operation. This tends to rate documents containing multiple words in the query higher. The formulas from Chapter 2 for the Extended Boolean AND operation are shown below:

$$w_{x,j} = f_{x,j} \bullet \frac{idf_x}{\max_i idf_i} \text{ and,}$$

$$sim(q_{and}, d) = 1 - \left(\frac{(1 - x_1)^p + (1 - x_2)^p + \dots + (1 - x_m)^p}{m} \right)^{\frac{1}{p}}$$

Where $x = W_{i,j}$

4.5 Tool Demonstration

This section begins with a presentation of the hardware and software used to develop the Hierarchical Thesaurus and Indexing Tool and the Query Tool. It also presents an overview of the databases and queries by which these tools were tested.

4.5.1 Hardware and Software Platforms

The two tools used in this research were developed in version 1.3 of the Java programming language. The system on which these tools were tested had the following hardware specifications:

- Intel Pentium III 550 MHz processor
- 384 MB RAM

4.5.2 Databases

The relational database used in the testing phase of this research was Microsoft Access 2000 from the Microsoft Office 2000 suite. It provided a mechanism to transform the .csv (comma separated) files into the desired tables with ease.

4.5.2.1 Hierarchical Thesaurus Testing Databases

The hierarchical thesaurus testing databases are comprised of three independent databases comprised of one table each. Each table contains 25 records, or documents, with simulated images as would be expected in the VDL environment. These documents were contrived due to the inability to gain access to the description field of the actual SDMS database within VDL. Each database contains an ID field to uniquely identify each record, a description field, an object field, squint direction field, turret rotation field, barrel elevation field, and camouflage percentage field. Figure 19 displays the hierarchies created to test the various characteristics of the thesaurus and stemming. Each of the three databases contains a different column heading to represent these fields.

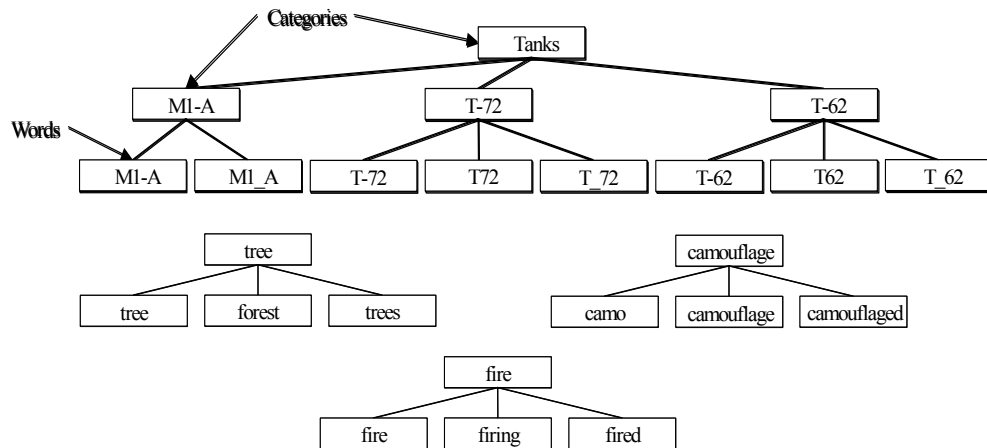


Figure 19. Created Hierarchies of Synonyms and Concepts for Testing

In addition, the primary focus of these images, T-72 and T-62 tanks are represented in several spellings in the databases to test the thesaurus tool. Another tank, the M1-A tank is included to test the hierarchical functionality of the thesaurus. These databases were also written in several verb tenses to test the stemming capabilities of this hierarchical approach.

4.5.2.2 Queries and Relevant Documents

A set of four queries was developed to simulate actual queries that could be submitted to the system. These queries were developed based on the knowledge of the contents of the databases. After query development, an expert user of the documents identified relevant documents that should be returned by these tools.

4.5.3 Evaluation Criteria

The criteria used to evaluate the performance of the query tool using the inverted file with the Hierarchical Thesaurus is discussed in the next few sections. Each of these

areas cover different aspects of the tools in this research with the most emphasis placed on the recall and precision measures.

4.5.3.1 Indexing Speed

The speed at which documents are indexed is not a truly important measure of the performance of the overall system. However, it is an important performance measure of the indexing tool by itself. For example, indexing a large document collection within a database should not take days to complete. This is important because the current setup requires the Hierarchical Thesaurus and Indexing Tool to not be in use for the Query Tool to have access to the B-Tree data structure for the information required to score documents.

4.5.3.2 Query Speed

The amount of time the Query Tool uses to return relevant documents is an important statistical measure. As the number of documents in the collection continues to grow, the amount of time to retrieve relevant documents will also grow. However, this growth in time should not be directly proportional to the size of growth of the document collection. Part of the solution to ensuring this does not occur was discussed previously. That is, not to score every document in the collection, but to only score documents that are in the occurrence list of the query words found in the inverted file. This can significantly decrease the amount of time to query. However, since no stop word lists were used and every word is indexed then the entire collection has the potential to be returned as relevant.

4.5.3.3 Relevant Document Retrieval

Relevant document retrieval is the most important measure of the effectiveness of the Hierarchical Thesaurus and the Query Tool. The primary goal of any tool returning relevant information is to put the most relevant information before the user first. When documents are returned, the documents that are most relevant to the user should be scored the highest and therefore be ranked higher on the returned documents list. There are two primary measures to determine the effectiveness of a relevant document retrieval tool, recall and precision. Each of these measures are discussed in the following sections.

4.5.3.4 Recall / Precision Variable Explanation

Figure 20 shows the Venn Diagram representation of precision and recall.

- $|R|$ = the number of documents in the entire document collection
- $|A|$ = the number of documents returned by the query tool
- $|Ra|$ = the number of documents returned and also relevant

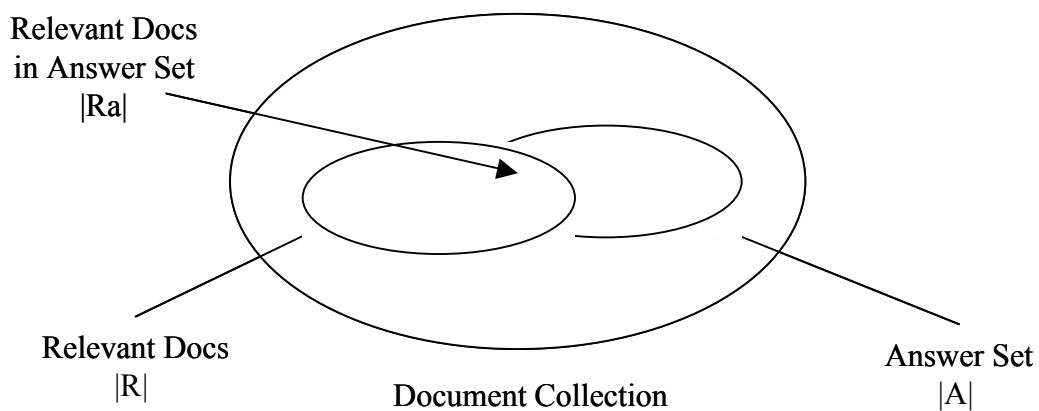


Figure 20. Variable Explanation for Precision and Recall

4.5.3.5 Precision vs. Recall

The measure of precision is a proportion of the retrieved documents that is relevant, while recall is a proportion of the relevant material retrieved. Both formulas can be seen below.

$$precision = \frac{|Ra|}{|A|} \qquad recall = \frac{|Ra|}{|R|}$$

Depending on their information needs, users may require a high recall, that is, the retrieval of any document that might be of interest. At other times, users may require a high precision, meaning the rejection of documents that are likely to be useless. Precision and recall values can change over time as the user examines more documents to determine their relevance. Typically, good information retrieval systems should maximize both a high precision and a high recall. It can be shown that the recall value increases as the number of documents returned increases, while precision decreases. Therefore, users interested in high recall tend to submit broader, more general queries, while users interested in higher precision tend to submit more specific queries [Salton, 83].

To determine the appropriate set of relevant documents, an expert in the area of the content of the documents must determine, given some query, which documents would be relevant and expected to be returned. To better understand the precision and recall measures an example is provided below. Assume the following

$$R_q = \{d_1, d_5, d_6\}$$

Where R_q is the set of relevant documents expected to be returned. The following documents were returned for some query q :

d₁ d₁ d₈ **d₅** d₃ **d₆**

The documents that are relevant to query q are bolded. To calculate the precision and recall measures, first examine the returned set of documents from top to bottom and see that the first document returned is relevant, d_1 . Since this document corresponds to 33% of all relevant documents then it can be said that we have a precision of 100% at 33% recall. Looking at the next relevant document, d_5 , it is found at position four, consequently, it would have a precision of 50% at 66% recall. Finally, the last relevant document would have 50% precision at 100% recall. This information from several algorithms can be plotted in a precision vs. recall graph to examine certain trends. Typically, the better performing algorithm's line will be closest to the upper right corner of the graph. The precision recall graph from the above example can be seen in Figure 21.

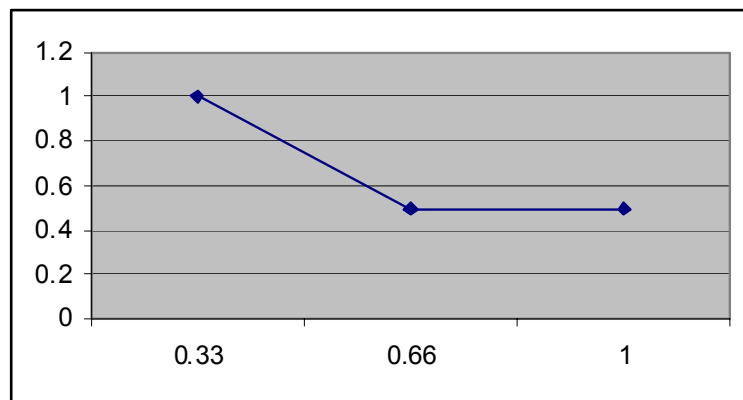


Figure 21. Precision-Recall Graph for Example

4.5.4 Results

4.5.4.1 Indexing Speed

The indexing speed was tested using various sizes of cache to limit the number of reads and writes to the B-Tree on disk. Results can be seen in Figure 22. Increasing the cache size from 1000 to 10,000 documents can effectively reduce the amount of indexing time by 300%. Note that the true indexing time, time spent not writing to disk, remains constant regardless of the cache size used.

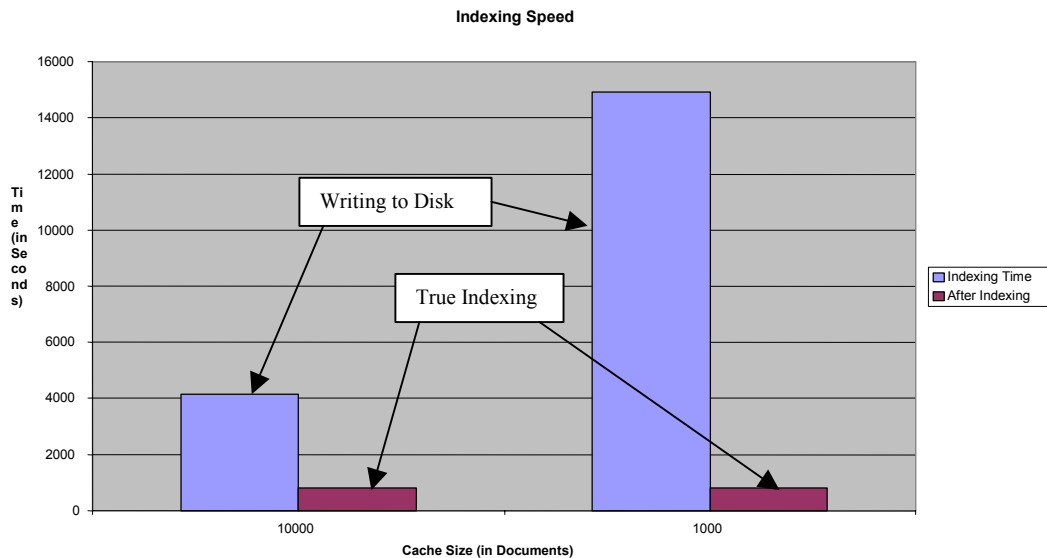


Figure 22. Indexing Speed with Various Levels of Caches

4.5.4.2 Query Speed

The query speed was tested on nine different databases ranging from 10,000 records to 90,000 records in 10,000 record increments. Identical queries were executed using the Vector and Extended Boolean model on each of the nine databases (Query

Terms: Development, Self, T-72, T-62). Variations of the Extended Boolean model were tried by varying the p-norm value from $p=1$ to $p=3$. The results can be seen in Figure 23.

As Figure 23 indicates, the query speed does grow linearly with the number of returned documents. This is the exact scenario the tool was trying to avoid. However, returning 90,000 documents to the user is unreasonable as most users would not want this many documents returned to search through to find their relevant documents. Therefore, users may take advantage of the filtering options of the query tool to narrow their search with more specific criteria.

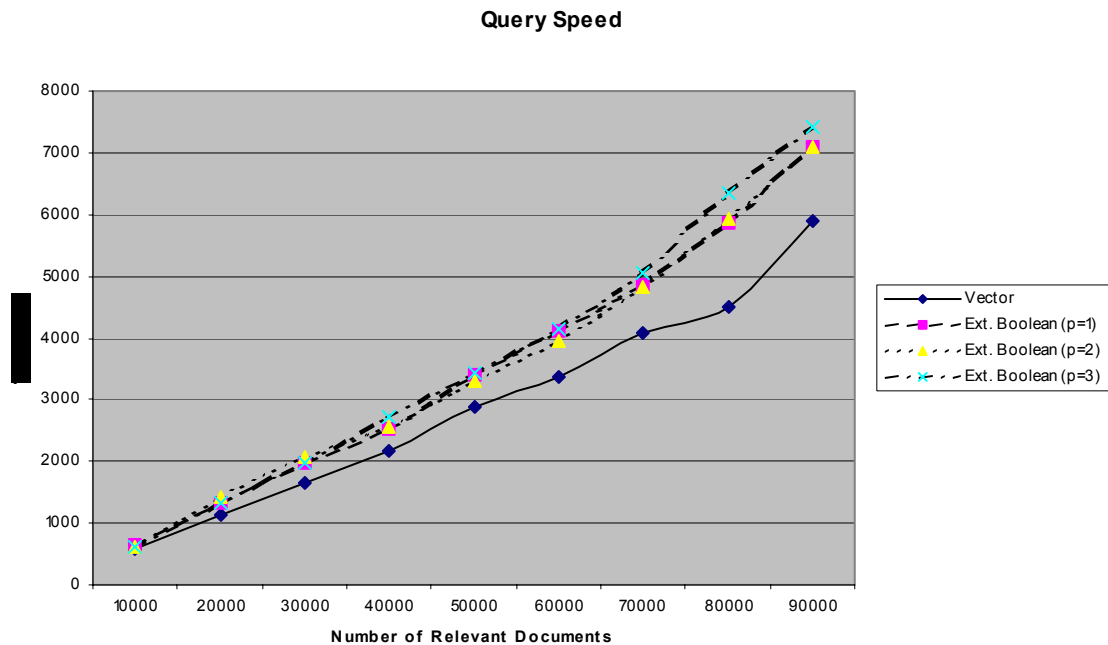


Figure 23. Query Speed of Various IR Models

4.5.4.3 Relevant Document Retrieval

The retrieval of relevant documents is the core function of the Query Tool. The results of the execution of the four queries discussed above can be seen in Figures 24 through Figure 27. Each individual query listed below was selected to test the hierarchical functionality of the thesaurus, but also the stemming approach discussed earlier. Without the thesaurus, the precision-recall graphs would go to zero, indicating that not all of the relevant documents have been returned, because no synonyms could be found within and across databases. Therefore, only a partial list of relevant documents would be retrieved depending on the spelling of the word entered. In addition, the test cases also indicate that the concept of a higher-level concept not found in the text of a document can also be located and scored. However, the ranking of the documents tend to favor the words that has the highest weights, however, all relevant documents were retrieved. In addition, the stemming approach appears to be a success. Without stemming, the precision-recall graphs of queries 1 and 2 would go to zero.

Queries:

1. Tank being fired upon
2. T-72 on bridge
3. Tank firing weapons
4. Tank with battle damage

However, there are some instances when the hierarchical thesaurus provided no increase in precision or recall. This is because the thesaurus cannot take into account phrases that may have the same meaning as an individual word. For instance, “battle damage” and “explosion” cannot be assigned synonyms of one another. This is why the precision-recall graph of query 4 is so poor. In addition, one must be careful when assigning words as synonyms of one another due to the possible multiple meanings of words.

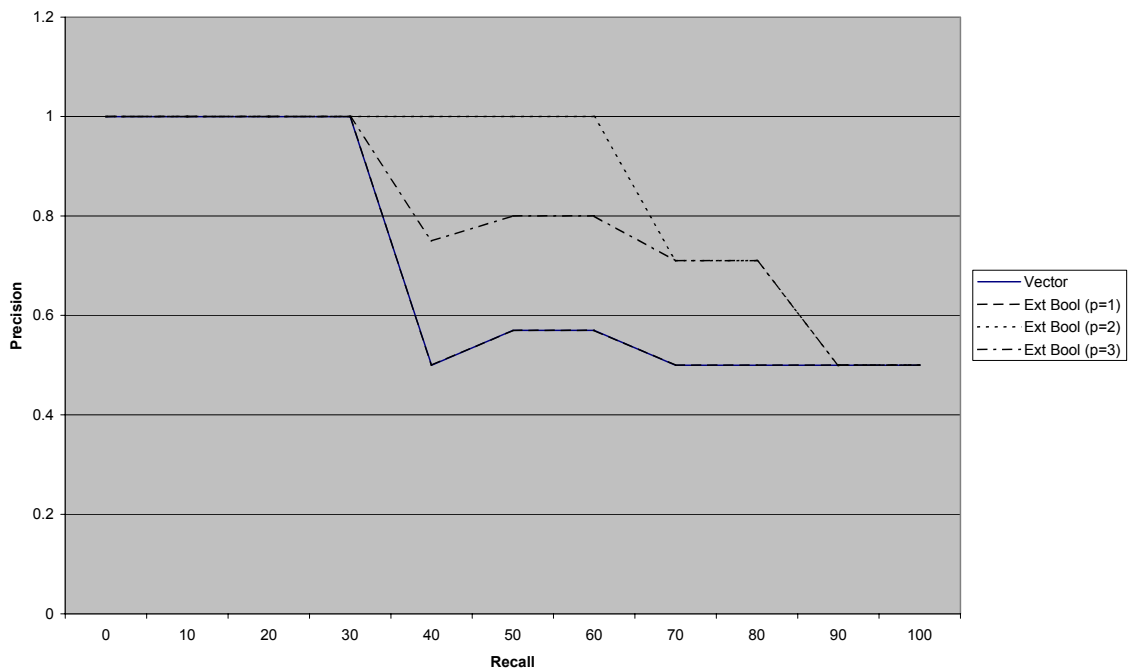


Figure 24. Precision-Recall Graph for Query 1

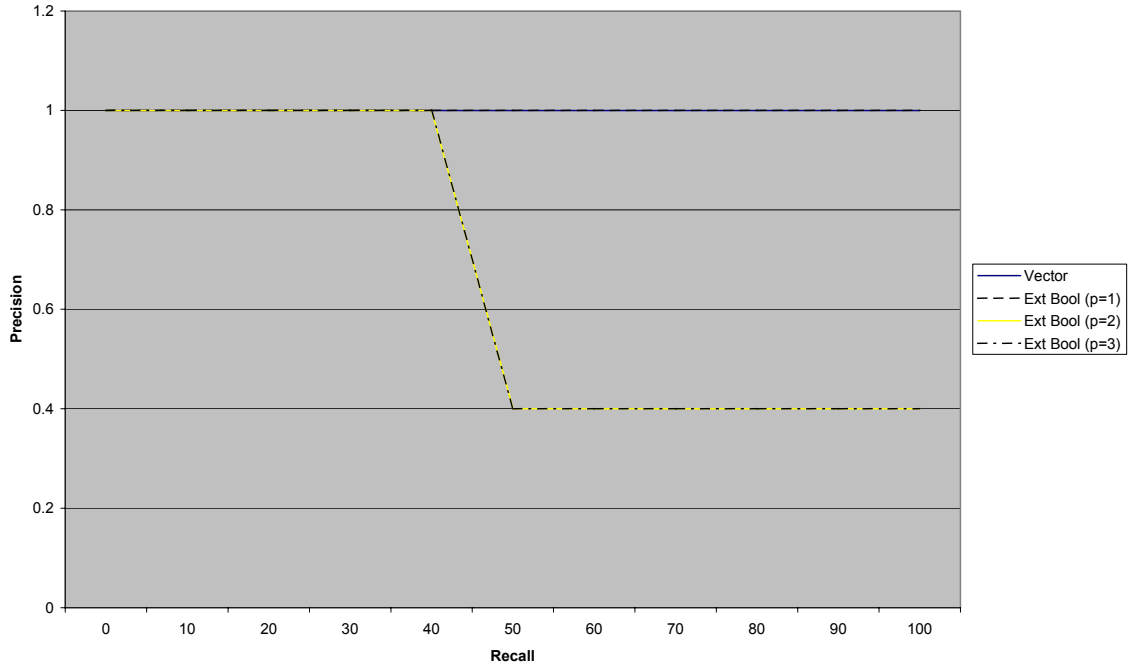


Figure 25. Precision-Recall Graph for Query 2

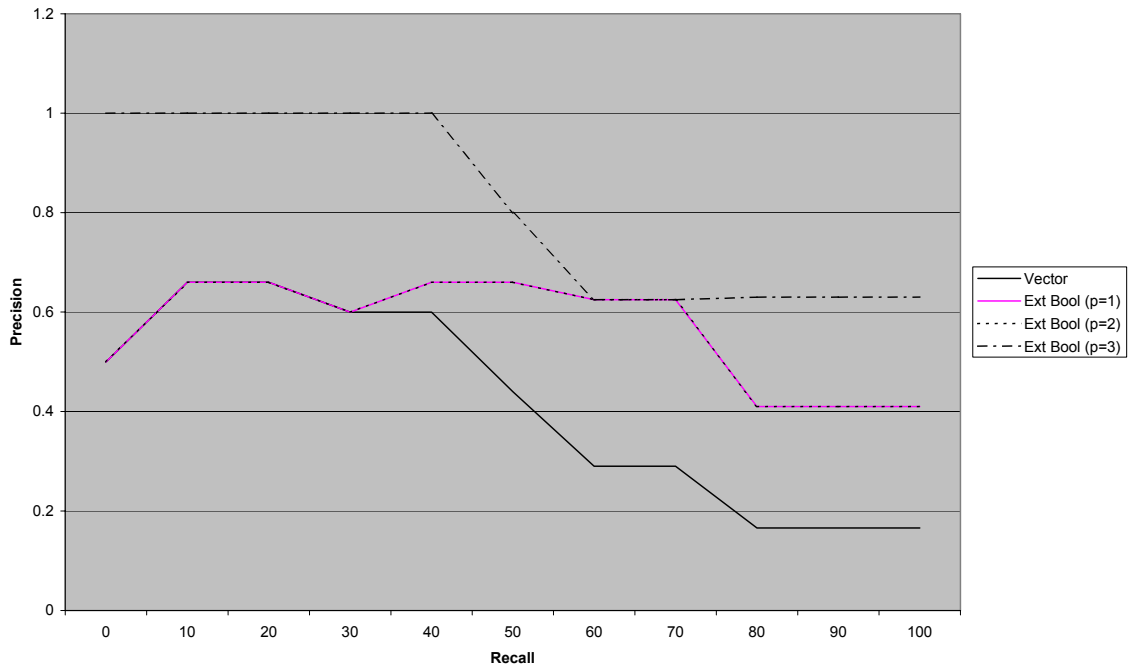


Figure 26. Precision-Recall Graph for Query 3

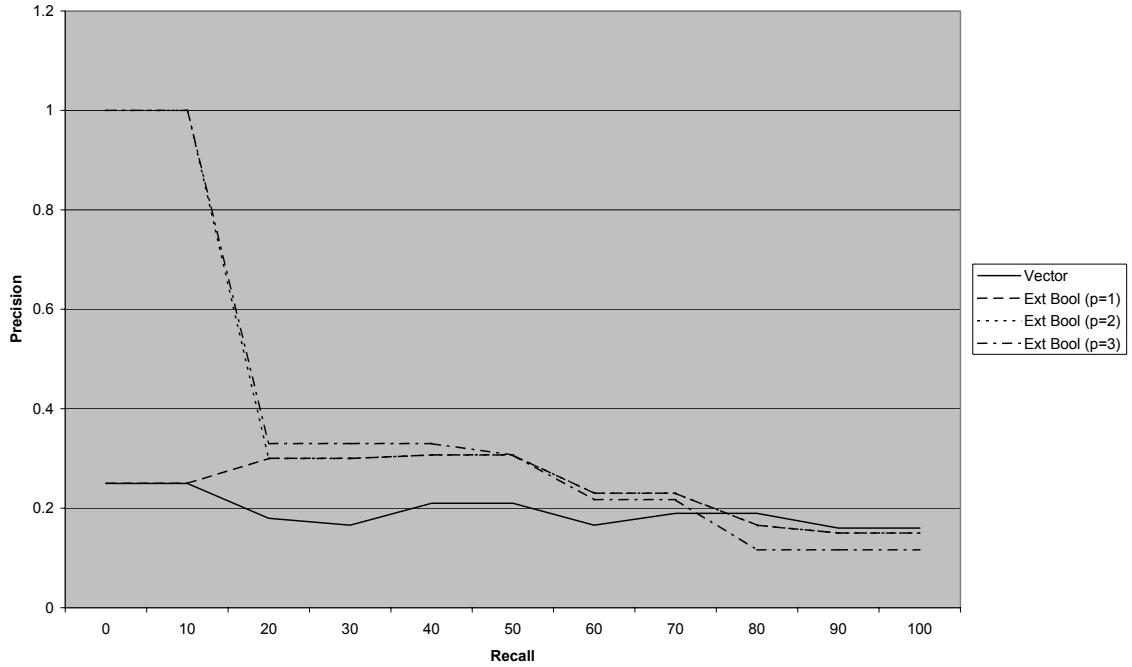


Figure 27. Precision-Recall Graph for Query 4

4.6 Summary

This chapter presents the implementation of the Hierarchical Thesaurus and Indexing Tool and Query Tool as outlined in Chapter 3. The implementation of the Schema Integration Tool and the Query Tool follow the guidelines of the functional requirements of each tool discussed in Chapter 3. During the implementation two key issues needed to be addressed, speed and memory. The precision and recall evaluation criteria are the primary determinate of how well these tools function together. Ultimately, the precision-recall graphs indicate there is a significant improvement in performance when implementing the hierarchical concept thesaurus methodology to retrieve relevant documents from heterogeneous databases.

V. Conclusions and Future Work

5.1 Summary of Research

This research has developed two tools, which together, provide users a mechanism to retrieve relevant documents from multiple heterogeneous databases within the VDL environment.

The first tool, the Hierarchical Thesaurus and Indexing Tool, has two major functional areas: hierarchical thesaurus creation and maintenance and document indexing. The hierarchical thesaurus provides the VDL administrator the capability to assign synonyms to words to alleviate the problems encountered within a multiple heterogeneous environment. This tool not only allows word synonyms, but also a hierarchy of categories, or concepts, that contain other categories or additional words. By taking this approach, stemming can also be accomplished, however without the space savings. The tool also performs the indexing functions to create an inverted file of words in the document collection and the documents in which they appear. The caching technique discussed in Chapter 3 and implemented in Chapter 4 provides increased speed when indexing a large number of documents in a database.

The Query Tool has two major functions as well. First the users may filter out undesired documents by specifying strict document filtering criteria before proceeding to the crucial step of retrieving relevant documents. The tool uses the inverted file created by the thesaurus tool to rank and return potential relevant documents.

5.2 Results

The initial results indicate the inclusion of a hierarchical thesaurus to map between various database schemas and data values significantly increase the precision-recall levels within the VDL environment is an overwhelming success. However, there are a number of areas that could be explored to possibly increase performance that are discussed in the next few sections.

5.3 Future Research Recommendations

Future research in information retrieval within the VDL area should focus on the incorporation of relevance feedback into the query tool. In addition, extending this tool to work in a web-based environment for multiple users, and the evaluation of the indexing and query capabilities to determine possible increases in speed and reduction of memory should also be evaluated.

5.3.1 Relevance Feedback

Extending the Query Tool to incorporate relevance feedback from the user should be the next step in increasing the functionality of the tool. This additional feature would allow a user to submit a second query with the characteristics of an ideal image that was retrieved by the initial query. This could be implemented using image thumbnails to allow the user to view the images that are the most relevant. This technique could be done any number of times by the user until they are satisfied with the returned results.

5.3.2 Phrase to Phrase and Phrase to Word Mapping

Currently the system implemented in this research is constrained by only mapping individual words to other individual words. The idea that multiple words mapping to either other multiple words or another single word should be explored to determine if this approach could increase system performance.

5.3.3 Web-based Application for Multiple Users

This research made the assumption that the databases, hierarchical thesaurus tool and query tool resided on the same computer. The next logical step is to extend the query tool to work in a web-based environment. This would allow multiple users from any where in the world to have access to the inverted file to retrieve relevant documents simultaneously.

5.3.4 Increased Indexing and Query Retrieval Speed

As seen in Chapter 4, the indexing speed is based on the size of the cache and the number of times the B-Tree is read and written to. Perhaps, other techniques, such as a file-based approach along with a merge sort would provide a faster indexing speed. In addition, the query speed increases directly proportional to the number of documents retrieved. Once again, possibly a different methodology could provide increased performance for processing documents during a query submission.

5.4 Summary

This work developed two tools, that when used together, can retrieve relevant documents from multiple heterogeneous relational databases. The foundation of this system is based on a hierarchical thesaurus by which synonyms and higher-level concepts can be assigned.

As a result of this work, AFRL/SN now has the capability to retrieve images from multiple databases, based on the metadata descriptions of the images found in the databases. These images are scored and ranked by order of relevance to the user by applying various information retrieval techniques.

The main focus of this research, the creation and maintenance of a hierarchical thesaurus has proven to dramatically increase the recall and precision by allowing users to submit synonyms and high-level concepts that may not be found directly in the document collection.

Recommendations for future research in this area include the application of user relevance feedback based on documents returned from the initial query, the development of a web-based implementation of this tool to allow multiple users at the same time, and increased efficiency in document indexing and relevant document retrieval regardless of document collection size.

In the future, the user can re-submit the query based on documents they have deemed relevant from the initial result set to pinpoint exactly what they are seeking.

A. Relevant Documents Retrieved

A.1 Query 1

Vector	Ext Bool 1	Ext Bool 2	Ext Bool 3
* 3000016	* 3000016	* 3000016	* 3000016
* 1000025	* 1000025	* 1000025	* 1000025
1000021	1000021	* 1000024	1000015
1000017	1000017	* 1000020	* 1000024
3000019	3000019	1000015	* 1000020
* 1000024	* 1000024	3000023	3000023
* 1000020	* 1000020	* 3000021	* 3000021
1000015	1000015	1000021	1000021
3000023	3000023	1000017	1000017
* 3000021	* 3000021	3000019	3000019
3000015	3000015	3000015	3000015
* 3000008	* 3000008	* 3000008	* 3000008

A.2 Query 2

Vector	Ext Bool 1	Ext Bool 2	Ext Bool 3
* 2000020	* 2000020	* 2000020	* 2000020
* 3000005	* 3000005	2000007	2000007
		2000015	2000015
		2000018	2000018
		* 3000005	* 3000005

A.3 Query 3

3000020	3000020	3000020	* 3000021
* 3000023	* 3000023	* 3000023	* 3000023
* 1000024	* 1000024	* 1000024	* 1000024
1000020	1000020	1000020	1000020
* 1000015	* 1000015	* 1000015	* 1000015
3000025	* 3000021	* 3000021	3000020
3000024	3000016	3000016	3000016
3000022	* 3000015	* 3000015	* 3000015
* 3000021	3000025	3000025	3000008
1000025	3000024	3000024	* 3000007
1000002	3000022	3000022	* 3000006
1000001	1000025	1000025	
3000019	1000002	1000002	
3000018	1000001	1000001	
3000017	3000008	3000008	
3000016	* 3000007	* 3000007	
* 3000015	* 3000006	* 3000006	

1000023
 1000022
 1000021
 1000019
 1000018
 1000017
 1000016
 1000014
 1000013
 2000008
 2000007
 2000005
 2000004
 2000003
 2000002
 2000001
 3000014
 3000013
 3000012
 3000011
 3000010
 3000009
 3000008
 * 3000007
 * 3000006

A.4 Query 4

Vector	Ext Bool 1	Ext Bool 2	Ext Bool 3
3000020	3000020	* 1000025	* 1000025
2000004	2000004	3000020	3000020
2000002	2000002	2000004	2000004
* 1000025	* 1000025	2000002	2000002
1000021	1000021	1000021	2000018
2000018	2000018	2000018	2000013
2000013	2000013	2000013	1000021
1000006	* 3000019	* 3000019	* 3000019
3000023	1000006	1000006	* 3000021
1000024	* 3000021	* 3000021	1000006
* 1000020	3000023	3000023	3000023
1000015	1000024	1000024	1000024
1000002	* 1000020	* 1000020	* 1000020
1000001	1000015	1000015	1000015
3000025	1000002	1000002	1000002
3000024	1000001	1000001	1000001
3000022	3000025	3000025	1000005
* 3000021	3000024	1000005	3000025

* 3000018	3000022	3000024	1000022
3000017	1000005	3000022	1000023
1000023	1000022	1000022	1000019
1000019	* 3000018	* 3000018	1000018
1000018	3000017	3000017	* 3000018
2000008	1000023	1000023	3000017
2000007	1000019	1000019	2000008
2000006	1000018	1000018	2000007
2000005	2000008	2000008	2000006
2000003	2000007	2000007	2000005
2000001	2000006	2000006	2000003
* 3000019	2000005	2000005	2000001
* 3000016	2000003	2000003	3000024
3000015	2000001	2000001	3000022
1000022	1000017	1000017	1000017
1000017	1000013	1000013	1000013
1000016	2000010	2000010	2000010
1000014	* 3000016	* 3000016	2000023
1000013	3000015	3000015	2000022
3000014	1000016	1000016	2000021
3000013	1000014	1000014	2000020
3000012	3000014	3000014	2000017
3000011	3000013	3000013	2000016
3000010	3000012	3000012	2000015
3000009	3000011	3000011	2000014
* 3000003	3000010	3000010	2000012
	3000009	3000009	2000009
	* 3000003	* 3000003	3000014
			3000013
			3000012
			3000011
			3000010
			3000009
			* 3000003
			3000002
			3000001
			1000012
			1000010
			1000007
			1000016
			1000014
			* 3000016

B. Query Speed

B.1 Query Speed for 10,000 Documents

10000	Vector	ExtBool 1	ExtBool 2	ExtBool 3
1	660	880	550	600
2	710	550	550	600
3	550	550	540	600
4	490	880	870	600
5	770	550	550	600
6	490	550	600	940
7	440	940	550	610
8	770	550	880	600
9	490	550	550	540
10	490	550	600	610
Average	586	655	624	630
StdDev	128.2532	169.8529	134.0149	110.755

B.2 Query Speed for 20,000 Documents

20000	Vector	ExtBool 1	ExtBool 2	ExtBool 3
1	1420	1590	1590	1210
2	990	1210	1380	1590
3	1260	1600	1590	1210
4	1370	1210	1210	1260
5	990	1650	1650	1600
6	1420	1150	1160	1200
7	990	1150	1700	1270
8	990	1480	1150	1650
9	930	1150	1210	1210
10	990	1150	1540	1210
Average	1135	1334	1418	1341
StdDev	205.6021	216.9588	219.5349	189.9386

B.3 Query Speed for 30,000 Documents

30000	Vector	ExtBool 1	ExtBool 2	ExtBool 3
1	1700	2470	2470	1920
2	1920	1870	1810	1980
3	1480	1810	1820	1920
4	1480	2080	2470	1920
5	1920	1810	1870	1920
6	1480	1870	1810	1980
7	1480	1810	2470	1930
8	2030	2250	1820	2360
9	1540	1870	1810	1920
10	1480	1870	2470	1920
Average	1651	1971	2082	1977
StdDev	223.4303	225.4107	334.3917	136.7926

B.4 Query Speed for 40,000 Documents

40000	Vector	ExtBool 1	ExtBool 2	ExtBool 3
1	2200	2860	2420	2580
2	1970	2360	2360	2530
3	2470	3130	2410	3070
4	1980	2360	3080	2470
5	2030	2420	2410	2580
6	2420	2800	2410	3080
7	1920	2360	2860	2580
8	2800	2360	2310	2520
9	1980	2360	2420	3180
10	1980	2360	2910	2530
Average	2175	2537	2559	2712
StdDev	294.4769	284.1772	277.3065	278.2006

B.5 Query Speed for 50,000 Documents

50000	Vector	ExtBool 1	ExtBool 2	ExtBool 3
1	2750	3070	3020	4060
2	3460	3790	3900	3240
3	2520	3020	3020	3180
4	3130	3790	3840	3130
5	2470	2970	3020	3130
6	3130	3740	3900	4120
7	2530	3020	2970	3180
8	3130	3790	2970	3180
9	2530	2970	3510	3960
10	3290	3900	3020	3130
Average	2894	3406	3317	3431
StdDev	372.6243	420.1904	419.5513	427.8486

B.6 Query Speed for 60,000 Documents

60000	Vector	ExtBool 1	ExtBool 2	ExtBool 3
1	3240	3740	3570	4890
2	3070	4450	4560	3790
3	3900	3630	3570	3790
4	3070	4510	4620	4510
5	3950	3630	3620	3790
6	3070	4670	3570	3790
7	4170	3620	4230	4440
8	3080	4720	3570	3790
9	3020	3570	4610	4890
10	3240	4560	3570	3850
Average	3381	4110	3949	4153
StdDev	443.0312	504.7772	490.5654	476.8892

B.7 Query Speed for 70,000 Documents

70000	Vector	ExtBool 1	ExtBool 2	ExtBool 3
1	3850	5390	5390	5440
2	3680	4390	4340	4620
3	4560	5280	5280	5710
4	3620	4340	4340	4560
5	4560	5330	5440	5550
6	3630	4340	4340	4560
7	4730	5490	5270	5500
8	3620	4340	4340	4560
9	4720	5380	5320	5430
10	3680	4340	4340	4610
Average	4065	4862	4840	5054
StdDev	504.3423	542.4185	529.2972	503.6798

B.8 Query Speed for 80,000 Documents

80000	Vector	ExtBool 1	ExtBool 2	ExtBool 3
1	550	5330	5980	6970
2	4450	6040	6040	6320
3	5330	6430	6480	6820
4	4450	5280	5280	5600
5	5110	5990	5980	6370
6	5600	6530	6480	6920
7	4440	5270	5270	5600
8	5160	5990	6040	6310
9	5600	6490	6540	6860
10	4450	5270	5330	5610
Average	4514	5862	5942	6338
StdDev	1473.026	532.3282	496.9865	564.1473

B.9 Query Speed for 90,000 Documents

90000	Vector	ExtBool 1	ExtBool 2	ExtBool 3
1	5160	7090	6970	7090
2	5830	7140	7200	7520
3	6150	7080	6970	7530
4	6040	7030	7140	7420
5	5880	7040	7190	7360
6	5930	7030	7030	7530
7	5980	7090	7200	7360
8	6040	7080	7140	7520
9	5870	7140	7090	7470
10	5980	7090	7090	7410
Average	5886	7081	7102	7421
StdDev	272.2825	39.5671	88.29244	134.5321

B.10 Summation of Query Speeds

	Vector	EB (P=1)	EB (P=2)	EB (P=3)
10000	586	655	624	630
20000	1135	1334	1418	1341
30000	1651	2082	2082	1977
40000	2175	2559	2559	2712
50000	2894	3317	3317	3431
60000	3381	3949	3949	4153
70000	4065	4840	4840	5054
80000	4514	5942	5942	6338
90000	5886	7102	7102	7421

Bibliography

Codd, E.F., "A Relational Model of Data for Large Shared Data Banks," CACM, 13(6): 377-387 (1970).

Cormen, T. and others. *Introduction to Algorithms* (14th Edition). Cambridge: The MIT Press, 1990.

Croft, W., and others. "Providing Government Information on the Internet: Experiences with THOMAS." Digital Libraries Conference DL '95: 19-24, (1995).

Dempsey, L. and R. Heery. "Specification for Resource Description Methods Part 1: A Review of Metadata: A Survey of Current Resource Description Formats." Report to the European Union on the DESIRE Project, 21 March 1997. <http://www.ukoln.ac.uk/metadata/desire/overview/>

Gilliland-Swetland, A. "Introduction to Metadata: Setting the Stage, Categorizing Metadata" Excerpt from Book. n pag. http://www.getty.edu/research/institute/standards/intrometadata/2_articles/index.html.

Harman, D. "How Effective is Suffixing?" The Journal of the American Society for Information Science, 42(1): 7-15, (1991).

Hooten, D.B. *A Traffic Pattern-Based Comparison of Bulk Image Request Response Time for a Virtual Distributed Laboratory*. MS thesis, AFIT/GCS/ENG/01M-03. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2001 (ADA392024).

Krovetz, R. "Viewing Morphology as an Inference Process." Computer science technical report. University of Massachusetts, Amherst: 93-96.

Mandala, R., and others. "Combining Multiple Evidence from Different Types of Thesaurus for Query Expansion." Proceedings of the 22nd Annual International ACM SIGIR conference on Research and Development in Information Retrieval, (1999).

Riloff, E. "Little Words Can Make a Big Difference for Text Classification." Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: 130-136, (1995).

Salton, G. and M. McGill. *Introduction to Modern Information Retrieval*. New York: McGraw-Hill, 1983.

Salton, G. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Englewood Cliffs: Prentice-Hall, 1971.

Salton, G. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computers*. New York: Addison-Wesley, 1989.

Salton, G. *Automatic Information Organization and Retrieval*. New York: McGraw-Hill, 1968.

Silberschatz, A. and others. *Database System Concepts* (4th Edition). New York: McGraw-Hill, 2002.

VDL White Paper, "Collaborative Environments: The Virtual Distributed Laboratory". Planet SDMS E-zine, Spring.

VDL Marketing Slides. Air Force Research Laboratory, Sensors Directorate. Wright-Patterson AFB, OH. 20 April 2001.

Ward, J. *Enhancing a Virtually Distributed Library User Interface Via Server-Side User Profile Caching*. MS thesis. AFIT/GCS/ENG/00M-23. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2000, (ADA380777).

Worsley, J. and J. Drake. *Practical PostgreSQL*. Command Prompt, Inc, 2001.
<http://stage.linuxports.com/projects/postgres/book1.htm>

Zobel, J., and others. "Inverted Files Versus Signature Files for Text Indexing." ACM Transactions on Database Systems, 23: 453-490 (December 1998).

Vita

First Lieutenant Rodney D. Lykins was born in 1976 in Dayton, OH. He graduated from Northmont High School in Dayton in June 1994. In 1998 he was accepted into the AFROTC at Bowling Green State University where he completed a Bachelor of Science in Computer Science and was commissioned in May 1998.

His first assignment as a Second Lieutenant was to the Communications Squadron, Vandenberg AFB, CA, as a Producer/Director in the Visual Information flight. In August 2000, he entered the Graduate School of Engineering and Management, Air Force Institute of Technology. Upon Graduation, he will be assigned to the 805th Computer Systems Squadron at Scott AFB, IL.

REPORT DOCUMENTATION PAGE

*Form Approved
OMB No. 0704-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) 05-03-2002	2. REPORT TYPE Master's Thesis	3. DATES COVERED (From - To) Mar 2001 - Mar 2002
--	--	--

4. TITLE AND SUBTITLE APPLICATION OF INFORMATION RETRIEVAL TECHNIQUES TO HETEROGENEOUS DATABASES IN THE VIRTUAL DISTRIBUTED LABORATORY	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S) Lykins, Rodney D., 1st Lieutenant, USAF	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFT/EN) 2950 P. Street, Building 640 WPAFB OH 45433-7765	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GCS/ENG/02M-06
---	--

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/SN (AFMC) Attn: Eric Baenen 5200 Springfield Pike, Suite 200 Dayton, OH 45431-1289 Comm: (937) 904-9309 Email: eric.baenen@vdl.af.mil	10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/SN
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

Maj. Karl S. Mathias, ENG, (937) 255-6565 x4280, Karl.Mathias@afit.edu

14. ABSTRACT
The Department of Defense (DoD) maintains thousands of Synthetic Aperture Radar (SAR), Infrared (IR), Hyper-Spectral intelligence imagery and Electro-Optical (EO) target signature data. These images are essential to evaluating and testing individual algorithm methodologies and development techniques within the Automatic Target Recognition (ATR) community. The Air Force Research Laboratory Sensors Directorate (AFRL/SN) has proposed the Virtual Distributed Laboratory (VDL) to maintain a central collection of the associated imagery metadata and a query mechanism to retrieve the desired imagery. All imagery metadata is stored in relational database format for access from agencies throughout the federal government. This research focuses on applying information retrieval techniques to the multiple heterogeneous imagery metadata databases to present users the most relevant images based on user defined search criteria. More specifically, it defines a hierarchical concept thesaurus development methodology to handle the complexities of heterogeneous databases and the application of two classic information retrieval models. The results indicate this type of thesaurus-based approach can significantly increase the precision and recall levels of retrieving relevant documents.

15. SUBJECT TERMS
Virtual Distributed Laboratory, Information Retrieval, Hierarchical Thesaurus, Vector Model, Extended Boolean Model

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 93	19a. NAME OF RESPONSIBLE PERSON Maj. Karl S. Mathias
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (Include area code) (937) 255-6565 x4280