13:54:54 OCA PAD AMENDMENT - PROJECT HEADER INFORMATION 01/12/96

Active

Project #: C-36-X21 Cost share #: Rev #: 6
Center # : 10/24-6-R7964-0A0 Center shr #: OCA file #:
Work type : RES
Contract#: F33615-93-1-1338 Mod #: P00003 Document : GRANT
Prime #: Contract entity: GTRC

Subprojects ? : N CFDA:
Main project #: PE #:

Project unit: COMPUTING Unit code: 02.010.300
Project director(s):
NAVATHE S B COMPUTING (404)894-3152

Sponsor/division names: AIR FORCE / WRIGHT-PATTERSON AFB, OH
Sponsor/division codes: 104 / 002

Award period: 930930 to 970131 (performance) 970331 (reports)

Sponsor amount	New this change	Total to date
Contract value	0.00	472,815.00
Funded	84,438.00	472,815.00
Cost sharing amount		0.00

Does subcontracting plan apply ?: N

Title: A KNOWLEDGE-BASED APPROACH TO INTEGRATING AND QUERYING DISTRIBUTED HETEROGEN.

PROJECT ADMINISTRATION DATA

OCA contact: Anita D. Rowland 894-4820

Sponsor technical contact

CHARLES SATTERTHWAITE
(513)255-6548

WL/AAAF-3
BUILDING 635
2185 AVIONICS CIRCLE
WRIGHT-PATTERSON AFB, OH 45433-7301

Sponsor issuing office

BILL BEEMAN
(513)255-6908

WL/AAKG
AREA B, BUILDING 7
2530 C STREET
WRIGHT-PATTERSON AFB, OH 45433-7607
FAX: (513) 255-3985

Security class (U,C,S,TS) : U ONR resident rep. is ACO (Y/N): Y
Defense priority rating : supplemental sheet
Equipment title vests with: Sponsor GIT X
SEE GRANT PARAGRAPHS I.15 AND I.16
Administrative comments -
MOD P-3 AWARDS THE FINAL INCREMENT OF FUNDS; CONTRACT IS FULLY FUNDED.

4
(5)

Closeout Notice Date 03-JUL-1997

Project Number C-36-X21 Doch Id 45412

Center Number 10/24-6-R7964-0A0

Project Director NAVATHE, SHAMKANT

Project Unit COMPUTING

Sponsor AIR FORCE/WRIGHT-PATTERSON AFB, OH

Division Id 3347

Contract Number F33615-93-1-1338 Contract Entity GTRC

Prime Contract Number

Title A KNOWLEDGE-BASED APPROACH TO INTEGRATING AND QUERYING DISTRIBUTED HETEROG

Effective Completion Date 31-JAN-1997 (Performance) 31-MAR-1997 (Reports)

Closeout Action:	Y/N	Date Submitted
Final Invoice or Copy of Final Invoice	Y	
Final Report of Inventions and/or Subcontracts	Y	
Government Property Inventory and Related Certificate	Y	
Classified Material Certificate	N	
Release and Assignment	N	
Other	N	

Comments

Distribution Required:

Project Director/Principal Investigator	Y
Research Administrative Network	Y
Accounting	Y
Research Security Department	N
Reports Coordinator	Y
Research Property Team	Y
Supply Services Department	Y
Georgia Tech Research Corporation	Y
Project File	Y

NOTE: Final Patent Questionnaire sent to PDPI

C-36-X21
n/a

Georgia Tech

Office of Grants and Contracts Accounting

Georgia Institute of Technology
190 Bobby Dodd Way
Atlanta, Georgia 30332-0259
USA
404•894•4624; 2629
Fax: 404•894•5519

January 22, 1996

AF PROGRAM MANAGER
Avionics Directorate Wright Laboratory
Attn: WL/AAF - 3- (Charles Satterthwarte)
2185 Avionics Circle, Bldg. 620
Wright- Patterson AFB, OH 45433-7301

Subject: Grant # F33615-93-1-1338

Dear Manager:

Enclosed is the Federal Cash Transactions Report (SF 272) and the Financial Status Report (SF269A) for the above noted grant covering the period January 1, 1995 through December 31, 1995.

If you have any questions or require additional information, please contact Kate Edwards at (404) 894-5522.

Sincerely,

David V. Welch
Director

DVW/ke

Enclosures

xc: C-36-X21/246R79640A0
Contracting Officer
ARPA/SSTO
WL/AAF
Wanda Simon, OCA, mailcode 0420

FINANCIAL STATUS REPORT

(SHORT FORM)

(Follow instructions on the back)

1. FEDERAL AGENCY AND ORGANIZATIONAL ELEMENT TO WHICH REPORT IS SUBMITTED	2. FEDERAL GRANT OR OTHER IDENTIFYING NUMBER ASSIGNED BY FEDERAL AGENCY	OMB Approval No.	PAGE	OF
U S AIR FORCE	F33615-93-1-1338	0348-0039	1	1 Pages

3. RECIPIENT ORGANIZATION (Name and complete address, including ZIP code)
GEORGIA TECH RESEARCH CORPORATION, P.O. BOX 100117, ATLANTA, GA 30384

4. EMPLOYER IDENTIFICATION NUMBER	5. RECIPIENT ACCOUNT NUMBER OR IDENTIFYING NUMBER	6. FINAL REPORT	7. BASIS
58-0603146	C-36-X21/246R79640A0	[ ] YES [X] NO	[X] CASH [ ] ACCRUAL

8. PROJECT/GRANT PERIOD (See instructions)		9. PERIOD COVERED BY THIS REPORT	
FROM: (Month, Day, Year)	TO: (Month, Day, Year)	FROM: (Month, Day, Year)	TO: (Month, Day, Year)
9/30/93	1/31/97	1/1/95	12/31/95

10. TRANSACTIONS:	I PREVIOUSLY REPORTED	II THIS PERIOD	III CUMULATIVE
a. Total Outlays	190,464.16	47,000.00	237,464.16
b. Recipient share of outlays	0.00	0.00	0.00
c. Federal share of outlays	190,464.16	167,558.96	358,023.12
d. Total unliquidated obligations			0.00
e. Recipient share of unliquidated obligations			0.00
f. Federal share of unliquidated obligations			21,665.05
g. Total Federal share (sum of lines c and f)			379,688.17
h. Total Federal funds authorized for this funding period			388,377.00
i. Unobligated balance of federal funds (Line h minus line g)			8,688.83

11. Indirect Expense	a. Type of Rate (Place "X" in appropriate box)							
	[X] Provisional		[ ] Predetermined		[ ] Final		[ ] Fixed	
	b. Rate		c. Base		d. Total Amount		e. Federal Share	
	FY94	37.00%	65,741.38		24,324.31		24,324.31	

12. Remarks: Attach any explanations deemed necessary or information required by Federal sponsoring agency in compliance with governing legislation.

FY95	40.00%	111,764.99	44,706.01	44,706.01
FY96	43.00%	77,962.54	33,523.89	33,523.89

13. Certification: I certify to the best of my knowledge and belief that this report is correct and complete and that all outlays and unliquidated obligations are for the purposes set forth in the award documents.

Typed or Printed Name and Title	Telephone (Area code, number and extension)
David V. Welch, Director Grants and Contracts Accounting	(404) 894-2629
Signature of Authorized Certifying Official	**Date Report Submitted**
	1/22/96

Questions concerning this report should be directed to Kate Edwards (404) 894-5522.

FEDERAL CASH TRANSACTIONS REPORT

Approved by Office of Management and Budget, No. 80-R0182

(See instructions on the back. If report is for more than one grant or assistance agreement, attach completed Standard Form 272-A.)

1. Federal sponsoring agency and organizational element to which this report is submitted

U S AIR FORCE

2. RECIPIENT ORGANIZATION

Name : GEORGIA TECH RESEARCH CORPORATION

Number and Street : P.O. BOX 100117

City, State and ZIP Code: ATLANTA, GA 30384

3. FEDERAL EMPLOYER IDENTIFICATION NO. 58-0603146

4. Federal grant or other identification number: F33615-93-1-1338

5. Recipient's account number or identifying number: C-36-X21/R79640A0

6. Letter of credit number: N/A

7. Last payment voucher number: N/A

Give total number for this period

8. Payment Vouchers credited to your account

9. Treasury checks received (whether or not deposited): 1

10. PERIOD COVERED BY THIS REPORT

FROM (Month, day, year): 1/1/95

TO (month, day, year): 12/31/95

11. STATUS OF FEDERAL CASH

(See specific instructions on the back)

a. Cash on hand beginning of reporting period	$ (44,632.98)
b. Letter of credit withdrawals	0.00
c. Treasury check payments	44,632.98
d. Total receipts (Sum of lines b and c)	44,632.98
e. Total cash available (sum of line a and d)	0.00
f. Gross disbursements	167,558.96
g. Federal share of program income	0.00
h. Net disbursements (Line f minum line g)	167,558.96
i. Adjustments of prior periods	0.00
j. Cash on hand end of period	$ (167,558.96)

12. THE AMOUNT SHOWN ON LINE 11J, ABOVE, PRESENTS CASH REQUIREMENTS FOR THE ENSUING DAYS

13. OTHER INFORMATION

a. Interest Income	$
b. Advances to subgrantees or subcontractors	$

14. REMARKS (Attach additional sheets of plain paper, if more space is required)

15. CERTIFICATION

I certify to the best of my knowledge and belief that this report is true in all respects and that all disbursements have been made for the purpose and conditions of the grant or agreement.

AUTHORIZED CERTIFYING OFFICIAL

SIGNATURE

DATE REPORT SUBMITTED: 1/22/96

TYPED OR PRINTED NAME AND TITLE: DAVID V. WELCH, DIRECTOR
Office of Grants and Contracts Accounting

TELEPHONE (Area code, Number, Extension): (404) 894 - 2629

THIS SPACE FOR AGENCY USE

272-102

STANDARD FORM 272 (7-76)
Prescribed by Office of Management and Budget
Cir. No. A-110

Questions concerning this report should be directed to Kate Edwards (404) 894-5522.

==========

=== PROJECT SUMMARY ===

ORGANIZATION: Georgia Institute of Technology

SUBCONTRACTORS: None

PRINCIPAL INVESTIGATORS:
Shamkant B. Navathe, Professor
email:sham@cc.gatech.edu
Phone: 404 853 0537
Fax 404 894 9442

TITLE OF EFFORT: Knowledge Based Query Processing

OBJECTIVE:
(A concise statement of what you are attempting to accomplish and why. At most a few sentences.)

We are trying to address various aspects of query processing in distributed heterogeneous environments with special emphasis on incorporating knowledge at different levels. The knowledge relates to information about the sources of data, their structure, their content, and their overall relevance to the problem at hand.

APPROACH:

Our present approach to the issue of incorporating knowledge into query processing and formulation can be broken down in three areas:

1. Query Formulation
2. Semantic and multiple Query Optimization
3. Incorporation of learning into the mediation task.

We discuss each area briefly below.

1. Query Formulation:

In the query formulation area
We are investigating issues involved in integrating multiple sources of semi-structured data like text documents. We are studying user interface and visualization techniques to let the user discover the ways in which data is organized. This allows the user to determine how meaningful the underlying information sources are. and to discover the potentially useful ones. Some of the methods being used are:

a. Use of thesaurus during the query formulation process to prompt the user with additional words related to the query.
b. Techniques to visualize the query results and compare them with query words.
c. feedback from the user at different levels of granularity (like clusters of documents, individual documents, parts of a document, phrases and words.

The preliminary implementation shows that the performance of Information Retrieval systems can be improved by providing the right set of interaction techniques and visualization schemes.

2. Semantic and Multiple Query Optimization:

Query optimization is a decision process that selects the best query evaluation stra
of this process can be improved by providing better information about the contents of the database (i.e., meta-data). Furthermore, the process itself can be improved by incorporating the semantics of the database and by considering global plans which optimize execution over a set of queries.

A Meta-Data View Graph (MVG) is a network for organizing and managing information about a database. The nodes of the network represent logical views of the database and contain information specific to the corresponding data set. Statistical information (e.g., selectivity factors) is used by the query optimizer to generate more accurate estimates of execution cost. Semantic information (e.g., integrity constraints) is used to transform a query into a set of semantically equivalent queries giving the selection process more plans to choose from. Finally, when given a set of queries, the MVG network can identify common subexpressions, the results of which can be computed once and shared among the set of queries.

3. Incorporation of learning into heterogeneous database mediation:

Larrge-scale integrated knowledge systems can be, and often are, opaque to their users. But if the knowledge organization and information processing in these systems is not transparent, then the user may not be comfortable in using the system or be confident of the results it produces. Three issues are considered in designing transparent knowledge systems: how to explain and illustrate the system's reasoning to a user, how to explain and justify its results, and how to enable the user to explore and navigate its knowledge base.

In particular, endowing the knowledge systems with Structure-Behavior-Function meta-models may provide useful answers to these questions. Structure-Behavior-Function (SBF) models typically have been used for representing knowledge of the functioning of physical devices. KRITIK, an integrated autonomous system, for example, uses such device models for conceptual design in engineering domains. CANAH-CHAB, an interactive learning environment, views Kritik itself as an abstract device, and represents its knowledge organization and information processing in the form of a SBF model. Canah-Chab uses this SBF system model for explaining Kritik's reasoning and justifying its results to a user, and for enabling the he plan is to use the above systems together with real large scale databases to enable designers to access information of a heterogeneous nature intelligently for solving their design problems.

PROGRESS:

In the first year of this effort so far we have dealt with independent problems related to the areas of query formulation, optimization, and learning in the context of design application. We are now embarking on making use of (from Linguistic Data COnsortium) produced from previous ARPA efforts. We will be developing an integrated prototype as described below.

PRODUCTS: NONE.

FY94 ACCOMPLISHMENTS:

1. Developed a prototype query formulation system for dealing with real-life text databases with visualization and user feedback techniques.

2. Integrated previous work on semantic and multiple query optimization into a single problem environment.

3. Produced new techniques for explaining answers, use of meta-data, and model-based reasoning in the context of query processing.

Current work focuses on interfacing the KRITIK and Canah-Chab design tools with external D/KBSs. To accomplish this, we are studying the possibility of utilizing/extending the tools constructed under the ARPA I3 initiative. For example, IDI provides the design system

a uniform access mechanism to a set of external component database systems. IDI and LIM will provide a basis for easily importing D/KBS information into the existing design system structures. KQML is being studied for use in remote access of D/KBSs to facilitate transparent access of data on an internetwork.

FY-95 PLANS:

Our main objective is to build and demonstrate an intelligent interface to a set of (possibly autonomous) information sources including structured databases, knowledge bases, and unstructured data. The approach we have selected involves development of a mediator which utilizes meta-knowledge of the underlying information stores to aid a user in "browsing" the data for relevant information. To demonstrate this technology, we intend to augment the capabilities of both an autonomous (KRITIK) and interactive (Canah-Chab) device design system by providing a mediated interface between the design system and a collection of data/knowledge based systems (D/KBS). The mediator should provide the following to the design system: (see figure 1).

1. A uniform access method and view of any D/KBSs with relevant information irrespective of individual information system design.

2. Meta-data query facilities allowing the design system to determine relevant information about component parameters, previous design specifications, device function descriptions, etc. The mediator may also take an active role in helping the design tool determine what information may be helpful.

3. The mediator design must maintain the separation of concerns of the device design system from the query system. This will allow reuse of the mediated query system for other intelligent tasks such as planning.

4. The mediator must make data location (local vs. remote) and data organization (relational, knowledge-base, rule system, etc) transparent to the design tool.

5. The mediator should provide easy import of external D/KBS information into existing design system data structures including the integration of completely autonomous systems with transparent access of data on an internetwork. Future work will focus on extending the mediator to construct a system which, in addition to presenting a unified view/access of disparate information sources, provides at least a rudimentary understanding of the information in external sources to aid the user in location relevant data.

TECHNOLOGY TRANSITION:

The technology transition efforts will start after we have developed the above prototype in FY95. Currently plans are under way to collaborate with IBM and Stanford in the use of databases as well as with the Xerox Design Center for use of design data.

PUBLICATIONS:

1. S. B. Navathe and M.J. Donahoo, " Towards Intelligent Integration of Heterogeneous Information Systems," Sixth Int. Hong Kong Computer Society Database Workshop on Database Reengineering and Interoperability, March 3-4, 1995.

2. A, Veerasamy, S. Hudson and S. B. Navathe, " Visual Interfaces for Text Information Retrieval Systems," Proc. of Visual Database Systems -3, an

IFIP WG 2.6 Workshop, Lausanne, Switzerland, March 1995.

DATE PREPARED (ORIGINAL REPORT TO SPONSOR): August 29, 1994.

(NOTE: This entire summary should not be more than about 120 lines in length.)

A Federated Database Environment for Device Design

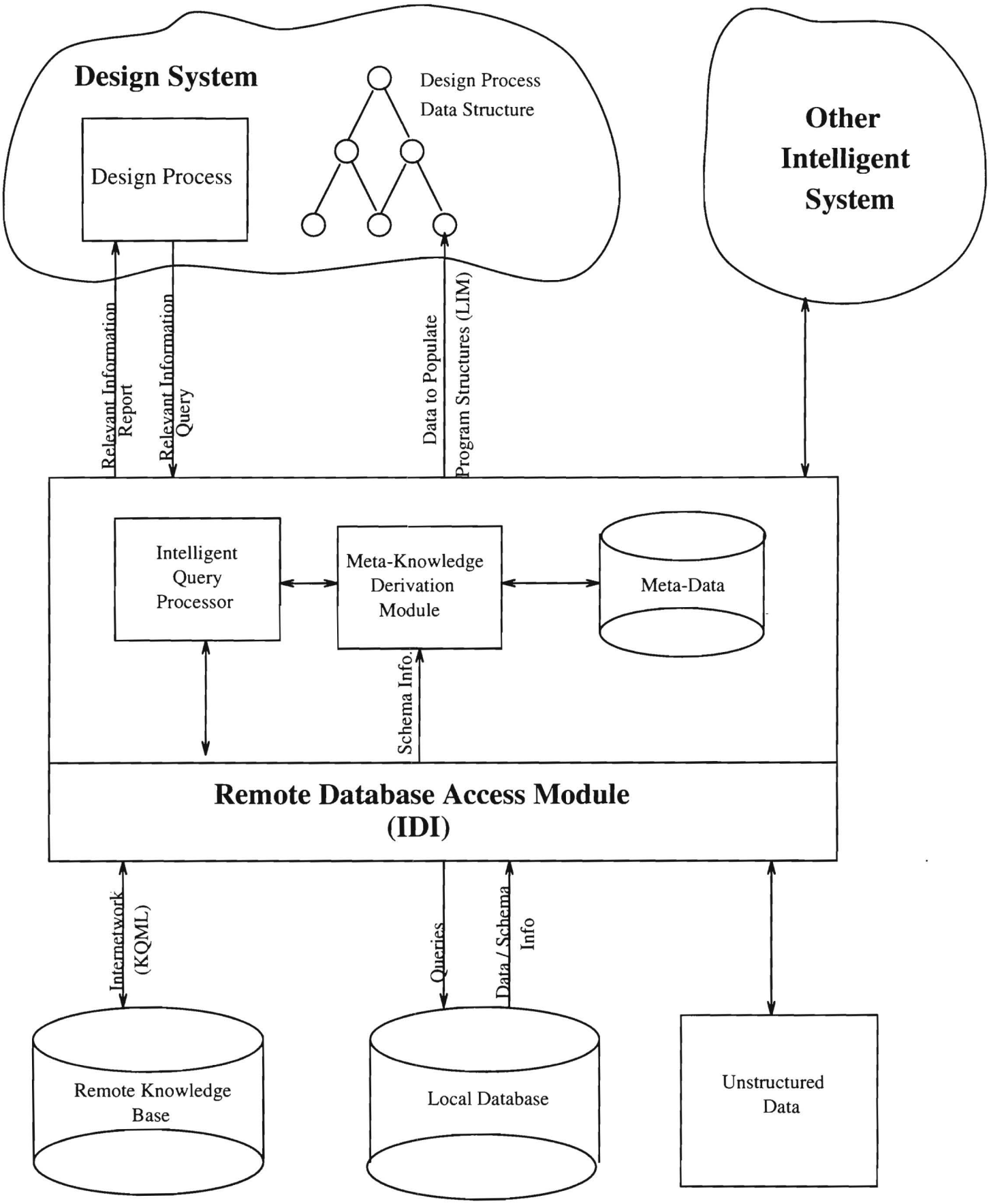


Figure 1: Proposed Architecture for an Intelligent Mediator for Device Design (Georgia Tech)

==========

=== ADMINISTRATIVE DATA ===

1. ARPA ORDER NUMBER: A522

2. CONTRACT/GRANT NUMBER: F33615-93-1-1338

3. AGENT: AIR FORCE - Wright Patterson Air Force Base, Ohio.

4. CONTRACT TITLE: Knowledge Based Query Processing
(official title: A Knowledge Based Approach to Integrating and Querying Distributed Heterogeneous Information Systems).

5. CONTRACTOR/ORGANIZATION: Georgia Institute of Technology

6. SUBCONTRACTORS: None

7. CO-PRINCIPAL INVESTIGATORS: None listed.

8. CONTRACT:

8.1. ACTUAL START DATE: September 30, 1993

8.2. EXPECTED END DATE: January 31, 1997.

9. FUNDING PROFILE:

9.1. Current contract: $472,815.

9.2. Total funds provided to date for all years $328, 377

Total funds expended to date for all years $143, 677

Report as of date July 31, 1994

9.3. Date total current funding will be expended: June 30, 1995.

9.4. Funds required in FY95:
(i.e., funds needed to fund you from date in 9.3 above through 11/30/95) at least $60K.

Note: The Government Fiscal year is 1 Oct - 30 Sep. Our intent is to fund through two months of the first Quarter of FY96 (i.e. 11/30/95).

10. ANYTHING ELSE YOU NEED (from ARPA):
Quick acquisition of software licensed under ARPA.

==========

===SIGNIFICANT EVENTS===

No events to report for FY 93.

==========

===PRESENTATION CHARTS===

To be supplied later in the required format.

==========

===WWW/MOSAIC HOME PAGE===

http://www.cc.gatech.edu/computing/Database/database.html

Towards Intelligent Integration of Heterogeneous Information Sources*†

Shamkant B. Navathe Michael J. Donahoo

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
{sham, mjd} @ cc.gatech.edu

Abstract

Current methodologies for information integration are inadequate for solving the problem of integration of large scale, distributed information sources (e.g. databases, free-form text, simulation etc). The existing approaches are either too restrictive and complicated as in the "federated" (global model) approach or do not provide the necessary functionality as in the "multidatabase" approach. We propose a hybrid approach combining the advantages of both the federated and multidatabase techniques which we believe provides the most feasible avenue for large scale integration. Under our architecture, the individual data site administrators provide an *augmented export schema* specifying knowledge about the sources of data (where data exists), their structure (underlying data model or file structure), their content (what data exists), and their relationships (how the data relates to other information in its domain). The augmented export schema from each information source provides an intelligent agent, called the "mediator," knowledge which can be used to infer information on some of the existing inter-system relationships. This knowledge can then be used to generate a partially integrated, global view of the data.

1 Introduction

Much of the research in database interoperability has focused on two extremes: multidatabase and federated systems. Multidatabase [Lit90, Spe88] systems provide a uniform access language to a set of database systems. While this is a necessary first step in solving the problems of heterogeneity, it places most of the integration responsibility on the user which may be unacceptable. Federated systems[She90] propose to create a global view of the underlying systems making the heterogeneity completely transparent to the user. While this approach is enticing, the complexity of constructing a global schema for large scale integration makes this approach infeasible because it requires an administrator who understands the semantics of all underlying systems and can resolve all inter-system schematic conflicts[Bat86]. In addition, the maintenance of a global schema in the face of addition/deletion of systems is difficult.

A better approach to interoperability involves the combination of techniques of reasoning and learning with techniques of data modeling and access to provide a partially integrated, global view. To accomplish this, the administrator of each underlying system presents a semantic description (augmented export schema) of their information to the "mediator." This augmented export schema may be as simple as the typical export schema or as detailed as a knowledge-based data description of the data, its relationships, and the system's domain. A knowledge-base system, such as Loom[Bri94], provides the capability to represent knowledge about the underlying information repositories and to

*Appeared in Proceedings of Sixth International Hong Kong Computer Society Workshop on Database Reengineering and Interoperability, Hong Kong, March 1995

†Both authors' work partially supported by ARPA Grant No. F33615-93-1-1338 under the Intelligent Integration of Information Program and Navathe's work partially supported by Army Research Office supported Center of Excellence in Information Science at Clark Atlanta University

make inferences as to the relationships among the various autonomous systems and generalizations concerning the information in each system. We have previously demonstrated that classification hierarchies can be effectively used to carry out integration of schemas[Sav91]. In this paper, we review the goals and strategy of the project HIPED, Heterogeneous Information Processing for Engineering Design, which we are currently pursuing at the Georgia Institute of Technology.

2 Related Work

Earlier work in integration provides the motivation and framework for our efforts. Batini et al. [Bat86] detail the problems of schema integration and provide a methodology for comparison of proposed solutions. Unlike many earlier integration efforts, we do not limit ourselves strictly to integration of databases. Instead, we focus on the integration of *information sources* including databases, free-form text, hypertext, etc. One possible method of dealing with this wide variety of information is to use Stanford's Object Exchange Model (OEM)[Pap94] which allows information exchange via *self-described* objects[Mar85] between different types of information sources. We propose to adapt the mediator paradigm[Pap94, Wei92, Wei93, Are94] to perform integration of the augmented export schemas. Integration of heterogeneous information sources requires a semantically rich data model. Earlier work has shown that the CANDIDE[Bec89, Nav91] model provides unique integration capabilities not found in traditional models. One major feature of the CANDIDE model is its ability to compute class-subclass relationships even among classes from dissimilar systems by subsumption from class relationship information[Sav91, She93, Wha93, Bra85]. Work with classification in the object-oriented model has produced similar results[Nav95, Are]. A variety of such systems supporting description logics are surveyed in [Bor94].

3 Approach

Our main objective is to build and demonstrate an intelligent interface to a set of (possibly autonomous) information sources including structured databases, knowledge bases, and unstructured data. Figure 1 shows our proposed architecture. The parenthetical references are made to applications developed under the ARPA I3 Initiative. KQML (Knowledge Query and Manipulation Language)[Cha92] allows remote access to knowledge/data bases. LIM (Loom Interface Module)[Par93b] allows import of external database information into Loom data structures. IDI (Intelligent Database Interface)[Par93a] is a common access language to several commercial database systems.

The approach we have selected involves development of an Engineering Design Mediator (EDM) which utilizes meta-knowledge of the underlying information to aid a user in "browsing" the data for relevant information sources and to make informed decisions about a plan for retrieving the appropriate data. To demonstrate this technology, we intend to augment the capabilities of both an autonomous (KRITIK2) and an interactive (Canah-Chab[Goe93]) device design system by providing a mediated interface between the design system and a collection of data/knowledge based systems (D/KBS). The mediator will be responsible for processing queries from the device design systems by determining where relevant data is, sending the appropriate query to the information site, performing the appropriate translations on the data, and returning the data to the design system. The design of the mediator is predicated on the following design goals:

1. Autonomy of the remote systems. Additionally, the remote systems should not be required to perform any functions outside of those defined for the internetwork connecting the system to the mediator.

2. Meta-data query facilities which allow the design system to determine relevant information about component parameters, previous design specifications, device function descriptions, etc.

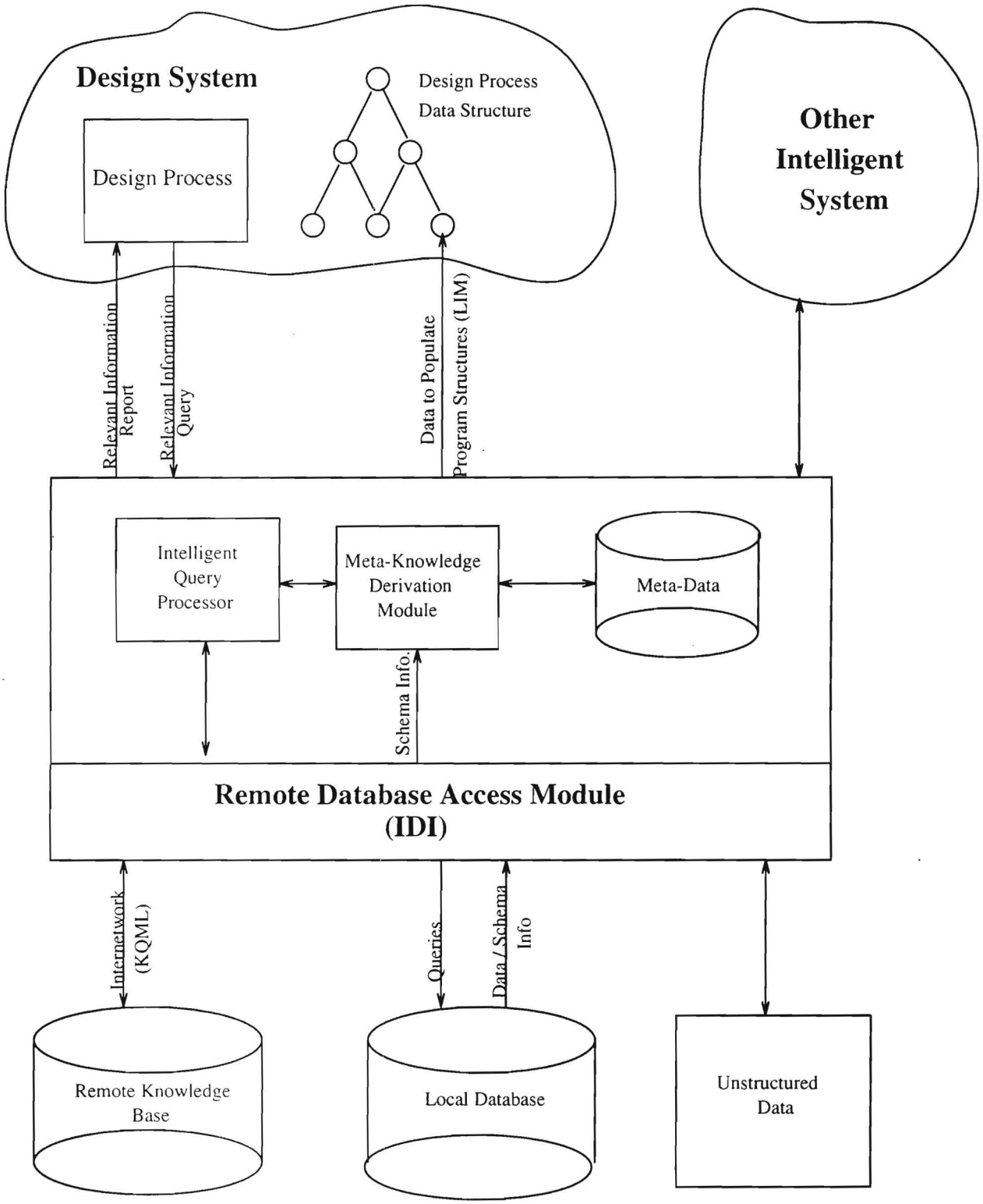


Fig. 1: Proposed Architecture for the Engineering Design Mediator (EDM)

The mediator may also take an active role in helping the design tool determine what information may be helpful (e.g. by use of a thesaurus, domain concept hierarchy, etc).

3. Separation of concerns of the device design system from the query system. This will facilitate reuse of the mediated query system for other intelligent tasks such as planning.

4. Data location (remote vs. local) and data organization (relational, knowledge base, text, etc) transparency.

5. Easy import of external D/KBS information into existing design system data structures minimizing the required changes to the device design system.

These constraints are designed to facilitate reuse of the mediator and to make the use of the system as transparent to intelligent applications as possible. Figure 2 presents an example query processing scenario.

4 Ongoing Research

Research is currently under way in the following areas to facilitate construction of a prototype query system which can be integrated with the device design system:

- Selection and development of the appropriate export data model to represent the data stored at each information source.
- Construction of an export knowledge model whereby information source administrators can express the relationships between their data and real world domain concepts. This in combination with the export data model will define the *augmented export schema.*
- Development of techniques for providing integration of the schemas of information sources into a partially integrated, global schema.
- Determination of optimization techniques for querying the remote information sources. Since the information sources may be interconnected with a WAN, a query processing bottleneck may arise with frequent remote data transmission.
- Provision of a query interface which aids the user in deriving the best answer to a query. Since no completely integrated schema exists and the user does not know what information is available, a query processor is required to guide users to the desired information.
- Capability of inferencing intersource knowledge from the augmented export schemas specifically concerning the relationships between information source entities.
- Ability to learn new, relevant knowledge about information sources based on user interaction.

5 Future Direction

Our initial focus is on providing access of integrated information to intelligent device design systems, but many other applications of this technology exist. With the advent of internetworks which connect thousands of computers all over the world, an explosion has resulted of the available data, both unstructured (text, graphical documents, audio, video, program sources) and structured (under DBMS control), accessible to hundreds of thousands of users. It would be difficult, if not impossible, to integrate all these sites with the current heterogeneous database techniques especially since most sites will not all be willing to provide services beyond those defined by the internetwork. Many query applications already exist for the Internet. WAIS servers provide keyword access to documents; however these documents must be under the control of a WAIS server. Gopher allows

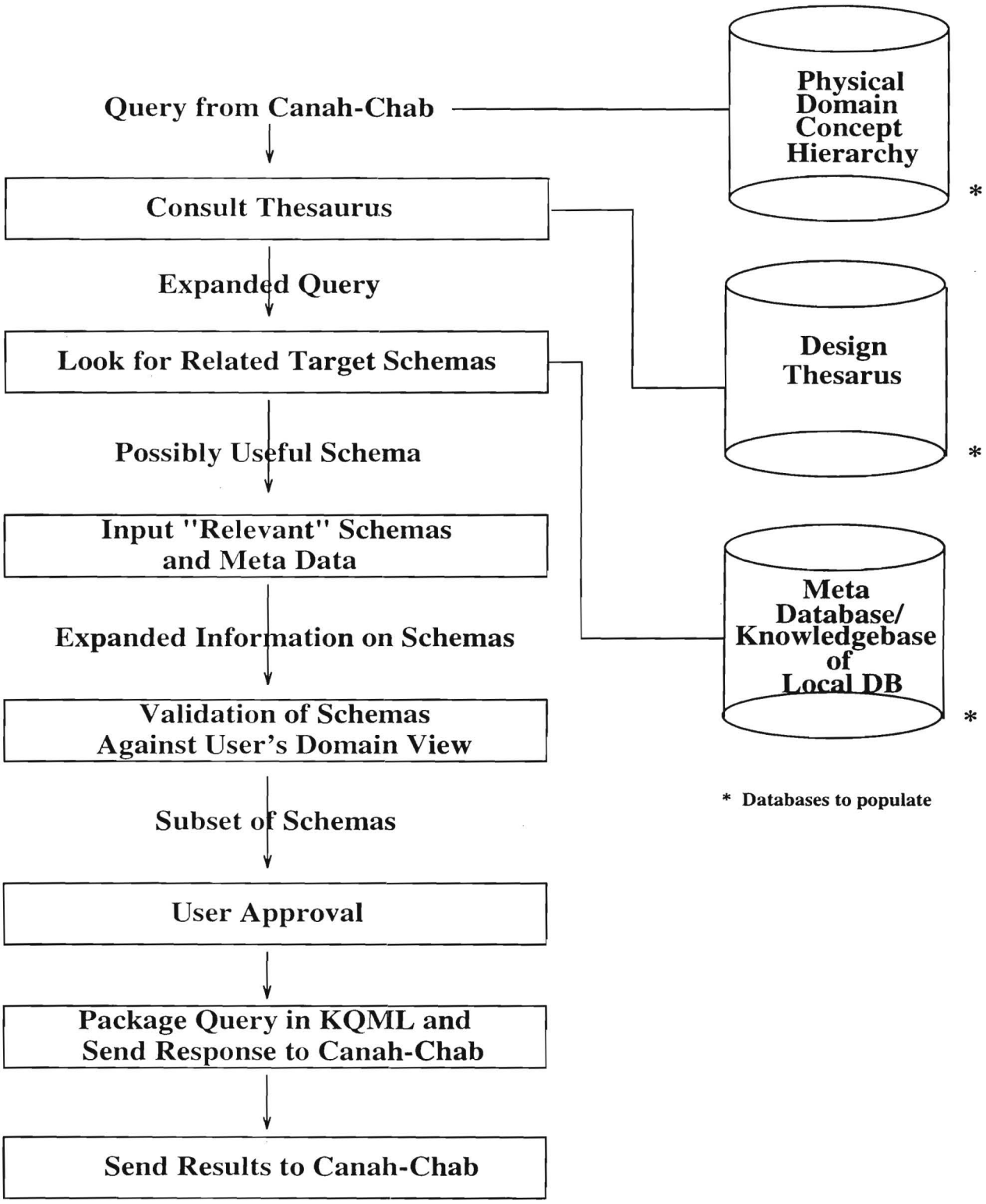


Fig. 2: Query Processing Scenario in the EDM

sites to setup directories of information that users can browse, but the information can only be accessed in the organization defined by the site manager. Archie provides a keyword query interface to find source code, but the keywords only work on the name of the source file (the user cannot ask for a program that performs some function, X; instead they must find the name of a program that performs X and search for it by name. World Wide Web (WWW) provides a nice interface to information organized by site managers (similar to gopher), but users suffer from the "hypertext navigation problem" which creates difficulties in locating specific information and keeping track of where they are in the web of hypertext documents over time.

Several problems exist for the tools mentioned above. First, the tools access a particular type of data (e.g. Archie only finds source code). If a manual exists for a particular application whose source code is found by Archie, the user is not informed. Second, the tools lack relativism because the users must access the data in the manner dictated by the site manager (e.g. in WWW the data is explicitly organized by hyperlinks). Third, some of the applications require a particular site organization (e.g. Gopher requires a specific directory structure). If a site has information but no desire to organize it, a gopher search may not find the relevant information at that site. Fourth, the query processors provide little organization to the data (e.g. Archie does not organize its source code references by application type, instead all applications with a substring match on the query are returned). For these reasons, the Internet environment provides a true testbed for large scale, heterogeneous information source integration.

We propose a query processing application which, using the native internetwork capabilities, provides a single interface for accessing all types of data regardless of source or format. The following list proposes some of the necessary extensions to the EDM:

- The system should perform automated "net surfing" to create an intelligent index of each data store's information. The intelligence of the index lies in the ability to discern between types of data (audio, text, source, etc), utilize an indexing methodology tailored to the particular data type (e.g. organize keywords of a text document by the document section), and facilitate determination of an object's relevance for a query based on the knowledge of the user's interests and technical expertise. This should require no a priori knowledge of the individual data site organization. Work is being done at the Georgia Institute of Technology in intelligent text document processing and work has been done at IBM Almaden Research Center in file classification[Vee95a]. Extensive work has been done on parsers for the various document types (e.g. html, LaTeX) on the Internet.

- The problem of data overload may result from this large scale integration. Our query processor should utilize user profiles so that only data of specific relevance and technical difficulty will be derived. Unfortunately, the user profile method of data overload reduction may eliminate relevant documents. To deal with this problem, the user needs feedback from the query processor in the form of a description of what information is/is not being considered and an explanation of why. Work in explanation is part of the Canah-Chab System[Goe93].

- Keyword searches should not be limited by the vocabulary of the query; instead, a thesaurus should be used to consider synonyms. This may result in synonym overload so user profiles should also be used in pruning the list of synonyms.

- The user is assumed to be "browsing" the available information; therefore, the query interface should provide reformulation capabilities. Reformulation techniques include iterative query alteration and positive/negative feedback from the user[Vee95b].

- The system should attempt automated knowledge acquisition to provide a better understanding of indexed objects and to find other available data stores. The following list orders levels of object knowledge in ascending complexity:

  ID Knowledge - System only knows site assigned ID of object (e.g. filename)

Content Knowledge - System knows information about object content (e.g. keywords for text)

Description Knowledge - System knows content knowledge and an external specification of the object.

Interrelational Knowledge - System knows all of the above and interobject relationships (e.g. papers about cancer research grouped together).

- The system should be extensible with respect to "plugging-in" different types of data indexing components and user profiles. Additionally, the system should transparently handle adding/subtracting participating sites. Utilities already exist for component indexing including parsers for various document types, image recognition utilities, etc.
- Different server systems should be able to exchange information and knowledge. Work in KQML at the University of Maryland facilitates knowledge interchange even with differing ontologies[Cha92].
- Objects must be described in terms of a nested model. For example, a document may be composed of sections which are composed of text, subsections, and graphics. Stanford's Object Exchange Model (OEM) provides "self-describing," nested objects[Pap94].
- The distributed control of the system leads to problems of object identity. For example, identical application source code may reside in multiple locations; therefore, the system should attempt to provide object identity to facilitate replicated object identification. Additionally, object versioning will allow the system to keep track of more recent versions of a retrieved object. A primitive form of object identification is supported in Stanford's OEM project [Pap94].
- External knowledge sources should be used to learn about objects in the system. For example, the query processor could inspect newsgroups or look at the manner in which objects are used in WWW to acquire knowledge about the objects and their relationships. Primitive forms of natural language understanding and concept derivation techniques may be used.
- Use of existing query systems should be considered (e.g. use WAIS server to augment search).
- Special consideration should be given to optimization including reuse of retrieved data[Don93].

6 Conclusion

We have presented a framework for research in the area of intelligent, large scale integration of information sources. Clearly, much more work needs to be done before any of the detailed functionality can be implemented. We believe that much of the research into the necessary technology has begun, and the main task lies in tailoring these technologies to the needs of large scale integration and applying them in a prototype environment. We intend to further study the concepts presented above in order to develop a flexible and extensible scheme for integrating information from heterogeneous sources. Although we wish to experiment by applying our research in the area of augmenting intelligent device design in engineering, the applicability of this technology obviously extends beyond the engineering domain.

References

[Are] Yigal Arens, Chin Chee, Chun-Nan Hsu, and Craig A. Knoblock. Retrieving and integration data from multiple information sources. To appear in International Journal on Intelligent and Cooperative Information Systems.

[Are94] Yigal Arens, Chin Chee, Chun-Nan Hsu, Hoh In, and Craig A. Knoblock. Query processing in an information mediator. ISI Technical Report, 1994.

[Bat86] C. Batini, M. Lenzernini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):325–364, Dec. 1986.

[Bec89] Howard W. Beck, Sunit K. Gala, and Shamkant B. Navathe. Classification as a query processing technique in the CANDIDE semantic data model. In *1989 IEEE Conference on Data Engineering*, pages 572–581. IEEE, 1989.

[Bor94] Alexander Borgida. Description logics in data management. Technical report, Rutgers University, July 1994.

[Bra85] R. Brachman and G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[Bri94] David Brill. *Loom Reference Manual (Version 2.0).* ISX Corp, October 1994.

[Cha92] Hans Chalupsky, Tim Finin, Rich Fritzson, Don McKay, Stu Shapiro, and Gio Weiderhold. An overview of KQML: A knowledge query and manipulation language. Technical report, KQML Advisory Group, April 1992.

[Don93] Michael J. Donahoo. *Integration of Information in Heterogeneous Library Information Systems.* Master's thesis, Baylor University, May 1993.

[Goe93] Ashok K. Goel, Andres Garza, Nathalie Grue, M. Recker, and T. Govindaraj. Beyond domain knowledge: Towards a computing environment for the learning of design strategies and skills. Technical report, College of Computing, Georgia Tech, 1993.

[Lit90] Witold Litwin, Leo Mark, and Nick Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22(3):267–293, September 1990.

[Mar85] Leo Mark. *Self-Describing Database Systems - Formalization and Realization.* PhD thesis, Computer Science Department, University of Maryland, 1985.

[Nav91] Shamkant Navathe, Sunit K. Gala, and Seong Geum. Application of the CANDIDE semantic data model for federations of information bases. In *Invited paper, COMAD '91*, Bombay, India, December 1991.

[Nav95] Shamkant B. Navathe and Ashoka N. Savasere. A practical schema integration facility using an object-oriented model. To be published in *Object Oriented Multidatabase Systems: A Solution for Advanced Applications* (O. Bukhres and A. Elmagarmid, eds), Prentice-Hall, January 1995.

[Pap94] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. Stanford University, Department of Computer Science, Technical Report, 1994.

[Par93a] Paramax System Corporation. *Computer System Operator's Manual for the Cache-Based Intelligent Data Interface of the Intelligent Database Interface*, revision 2.3 edition, Feb. 1993.

[Par93b] Paramax Systems Corporation. *Software Design Document for the Loom Interface Module (LIM) of the Cache-Based Intelligent Database Interface*, revision 2.0 edition, Jan. 1993.

[Sav91] Ashoka Savasere, Amit Sheth, Sunit Gala, Shamkant Navathe, and Howard Marcus. On applying classification to schema integration. In *First International Workshop on Interoperability in Multidatabase Systems*, pages 258–261. IEEE Computer Society, IEEE Computer Society Press, April 1991.

[She90] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[She93] Amit P. Sheth, Sunit K. Gala, and Shamkant B. Navathe. On automatic reasoning for schema integration. *International Journal of Intelligent and Cooperative Information Systems*, 2(1):23–50, 1993.

[Spe88] R. Speth, editor. *Global View Definition and Multidatabase Languages - Two Approaches to Database Integration.* Amsterdam: Holland, April 1988.

[Vee95a] Aravindan Veerasamy, Scott Hudson, and Shamkant Navathe. Visual interface for textual information retrieval systems. To appear in Proceedings of IFIP 2.6 Third Working Conference on Visual Database Systems, Lausanne, Switzerland, Springer Verlag, March 1995.

[Vee95b] Aravindan Veerasamy and Shamkant Navathe. Querying, navigating and visualizing an online library catalog. Submitted for Publication, January 1995.

[Wei92] Gio Weiderhold. Mediators in the architecture of future information systems. *IEEE Computer*, pages 38–49, March 1992.

[Wei93] Gio Weiderhold. Intelligent integration of information. In Arie Segev, editor, *ACM SIGMOD International Conference*, volume 22, pages 434–437. ACM, ACM Press, June 1993.

[Wha93] Whan-Kyu Whang, Sharma Chakravathy, and Shamkant B. Navathe. Heterogeneous databases: Toward merging and querying component schema. *Computing Systems*, 6(3), August 1993. (a Univ. of California Press publication).

Visual Interface for Textual Information Retrieval Systems[1][2]

Aravindan Veerasamy
Scott Hudson
Shamkant Navathe
{veerasam, hudson, sham}@cc.gatech.edu
College of Computing
801, Atlantic Drive
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

Abstract

A prototype user interface implementation for text information retrieval system is described. Using a visualization scheme, the interface provides visual feedback to the user about how the query words influence the ranking of retrieved documents. The interface also helps the user in constructing complex structured queries by simple drag-and-drop operations. An intuitive model where the user classifies the information provided to him/her as being positive and negative aids him/her in supplying rich relevance feedback information to the system. Our prototype interface has been built on top of INQUERY [CCH92]. Preliminary experience with the interface shows it to be a valuable tool in aiding the interactive search process between the user and the system. To test the effectiveness of the interface, we plan to conduct studies on users with real information need searching a large corpus of articles.

Keywords: Visualization of results, visual query languages, query processing, information retrieval

---

[1]This work was supported in part by ARPA Grant No. F33615-93-1-1338 under the Intelligent Integration of Information Program

[2]Appeared in the Proceedings of the Third Conference on Visual Database System, IFIP 2.6., Lausanne, Switzerland, March 1995

1 User Interface issues for Information Retrieval systems

User Interface issues and interaction techniques for full text information retrieval systems have in general received much less attention than system issues like document representation and retrieval algorithms. We have developed an interface that facilitates the user in visually constructing powerful queries for ranked output retrieval systems. The interface includes a scheme for visualizing the query results in a form that enables the user to see the relationships between the query results and the query. While a majority of online library catalog systems use a boolean model of retrieval, a vast majority of existing experimental information retrieval systems retrieve a ranked set of documents in decreasing order of relevance in response to a free-form textual query. In ranked output systems, the documents and the queries are modeled by a set of weighted index terms. The index term weighting function for the documents primarily takes into consideration

- the frequency of occurrence of the index term in the document,
- the number of documents in the corpus containing that index term.

The effectiveness of a retrieval system is measured by two metrics: recall (the ratio of the number of relevant documents retrieved to the total number of relevant documents in the corpus) and precision (the ratio of the number of relevant documents retrieved to the total number of documents retrieved). The reader is referred to [BC87, Rij79, SM83] for a comprehensive description of evaluation metrics of information retrieval systems, document representation and retrieval techniques.

While processing a free-form textual query, most ranked output Information Retrieval systems automatically extract index terms from the query and weight them. The weighted query index terms are then matched against the weighted index terms of documents to retrieve a ranked set of documents in decreasing order of relevance. Each document is weighted, the higher the weight of a document, the more likely it is to be relevant to the query. Most of the existing library information systems (On-line Public Access Catalogs, OPAC) follow a boolean retrieval model. In this model, the documents retrieved in response to a boolean query are not ranked. If a document satisfies the boolean query specification, it is retrieved. Compared to boolean systems, ranked output systems are a significant improvement since the query can be in a free-form text as opposed to a strict boolean syntax. Also, the retrieved documents are ranked, thereby placing the more useful documents at the top of the list. This is a particularly useful feature since it has been shown that users of boolean

systems spend a considerable effort in reducing the size of the result set [Spi93]. On the other hand, ranked output systems introduce a new problem: For a naive user, the logic behind the ranking of documents in response to a query is not as apparent and straightforward as a boolean system. The interface we have developed is aimed at alleviating this problem. It helps the user in understanding how the system computed the ranking of retrieved documents by visualizing the relationship between query terms and the results of the query.

The interface also aids the user in formulating complex structured queries by graphically manipulating objects on the screen. A simple mechanism of classifying any information on the screen into positive and negative instances lends itself to easy formulation of structured queries. The interface is built on top of INQUERY [CCH92], a ranked output retrieval system based on Bayesian inference networks. The interface supports two types of feedback:

- feedback from the user to the system and
- feedback from the system to the user.

It is interesting to note that the term "feedback" in the field of Information Retrieval typically refers to user's feedback to the system, while in the field of Human Computer Interfaces, "feedback" usually refers to the system's feedback to the user. The user's feedback to the system and the different levels of granularity at which the feedback can be provided is discussed in section 4. The system's feedback to the user and the visualization technique is discussed in section 5.

2 Related Work

Numerous studies on user interaction with online library access catalog systems with a boolean retrieval model have been conducted [Spi93, SS92, Dal90, Fid91a, Fid91b, Fid91c]. Spink [Spi93] studies the different forms of user feedback during a retrieval session. Of the total number of feedback actions by the user, 45% were aimed at adjusting the size of the retrieved set of documents, and about 40% were related to relevancy of documents. Fidel [Fid91a, Fid91b, Fid91c] discusses the issue of user interaction by studying the process of search term selection and searching styles in online library access catalogs. Dalrymple [Dal90] looks at the feedback process from a user-centered perspective. Bates [Bat] describes a boolean retrieval system which integrates an online thesaurus. None of the above studies involve a ranked output system supporting free-form textual queries. All of the systems deal with boolean

retrieval model only. We believe that there is a significant difference in the way users interact with a boolean system and a ranked output system. The reader is referred to [Har92] and [HB92] for a comparative discussion of boolean systems and ranked output systems. While building our interface we have borrowed valuable ideas from the studies mentioned above. In particular, the need to integrate an on-line thesaurus with the search interface in an easy-to-use fashion and a simple interaction scheme to include words from documents into the query have been influenced by the results of above-mentioned studies.

Walker and Beaulieu [Wal87, HB92] describe their OKAPI system which is a ranked output retrieval system for library catalogs. Similarly, Fox [FFS+93] describes their MARIAN sysem which is also a ranked output system for library catalogs based on the vector-space model. While OKAPI has facilities for relevance feedback and query expansion using a thesaurus, it largely lacks any means of providing system feedback to the user about how the ranking was computed. The interface we have developed integrates relevance feedback information from the user as well as feedback from the system illustrating the relationship between query results and query words.

A number of visualization schemes for information retrieval systems have also been proposed. The perspective wall [CRM91] describes a visualization scheme which supports browsing of documents. While such a system will not handle qualitative document classifications such as library subject catalogs, it is very useful for visualizing documents based on data which is linear in nature (like date of publication). Other visualization schemes such as [Kor91, Spo94, HKW94] have facilities for viewing a large document space. But visualizing the document space along more than 3 - 4 dimensions simultaneously becomes very cumbersome using the above systems. Also, most of them do not provide support for querying with relevance feedback and none of them provide support for query expansion using a thesaurus. The visualization scheme in our interface can gracefully handle much higher number of query word dimensions.

2.1 Novelty of our approach

The novelty of our system is in integrating a diverse set of interaction features in a seamless fashion into a single system thereby facilitating the interactive and iterative nature of the information seeking process. The following features are integrated in our system:

- Using a visualization scheme, the interface provides visual feedback to the user about how the query words influence the ranking of retrieved documents.

- By simple drag-and-drop operations of objects on the screen, the interface facilitates a naive end-user in constructing complex structured queries and in providing relevance feedback. This feedback is utilized by the system in a manner described later.

- The interface integrates an online thesaurus which provides words related to the query that can be used by the user to expand the original query.

Belkin and his group's work [BMC93, BMA+91, HB94] on user interfaces for information retrieval systems elucidates the issues in user interface and interaction techniques for full text retrieval systems. Belkin [BMA+91] mentions that "This type of analysis led to another important conclusion, namely that information systems for end users must support a variety of goals and tasks, but through some common interface or seamless access mechanism to a variety of relevant information sources and system functionalities". Our interface takes a step in that direction by integrating different pieces of information with a visualization scheme and simple interaction techniques.

3 Interactive Construction of Queries

Searching a database for information is a highly interactive process with the user constantly refining the query after examining the results of previous iteration until he/she is either satisfied with the results or is frustrated with the process and gives up. In existing information retrieval systems, the interaction proceeds by the user providing feedback on which of the retrieved documents are relevant to his/her information need. The system uses this information to modify the original query resulting in an improved ranking of retrieved documents. It has also been shown by Spink [SS92] that during iterative query reformulation, users tend to expand the query using search terms from various sources such as a thesaurus, previously retrieved documents and user's background knowledge. Expanding the query with terms from such sources can contribute to retrieval of more relevant documents in the next iteration.

Our interface encourages the interaction between the user and the system by providing the user with simple interaction technique to let him/her supply relevance feedback at different levels of granularity: whole documents, document portions, phrases and individual words. Almost any information appearing on the screen can be used for feedback. This is achieved by simple "drag-and-drop"ping the feedback object into either a "Positive Objects" window colored green or a "Negative Objects" window colored red. This scheme provides a simple abstraction to the user for classifying

any type of information without having to worry about what action to take for what type of information. A typical user session along with the response of the interface for every user action is described below using an example (please refer to Figure 1). The database being queried contains a collection of titles, authors and abstracts of thousands of CACM articles.

- The user types in his free form textual query in the query window. In the example shown in figure 1, the query is "image audio and text data compression".

- As every query word is typed in, the system consults an on-line thesaurus and displays words and phrases related to the query word in an adjacent window.

- At any point during the session the user can drag-and-drop any of the related words/phrases into the positive and negative windows. Internally the system expands the query by treating the positive words/phrases as synonyms of the corresponding query word. The negative words/phrases are included in the query with a NOT operator. For example, if for a query word "bank", the phrase "financial institution" is classified as positive and "river bed" is classified as negative, the corresponding internal query would be "#SYNONYM(bank #2[3](financial institution)) #NOT(#2(river bed))". The end-result of this classification is a possible improvement in the precision measure since documents containing the phrase "river bed" will be weighted lower than other documents, and a possible improvement in the recall measure since documents containing the phrase "financial institution" are also retrieved. The interface facilitates construction of such structured queries by simple drag-and-drop operations. In the example in figure 1, three words related to the query word "compression", namely, "compaction, "shortening" and "condensation" have been classified as positive. Internally the systems treats these three words as synonyms of "compression".

- After the user types in the query, the system evaluates the query and displays the titles of top-ranked documents in the "Query Results" window.

- The user examines the query result. Double-clicking any title with the mouse will bring up the full document.

- The user can classify any document as being relevant or non-relevant by drag-and-drop'ing the document into positive and negative windows. In the example in figure 1, the user has classified three documents titled "Experiments in text

[3] #2 is the proximity operator in INQUERY specifying that the words should appear within a distance of 2 within each other

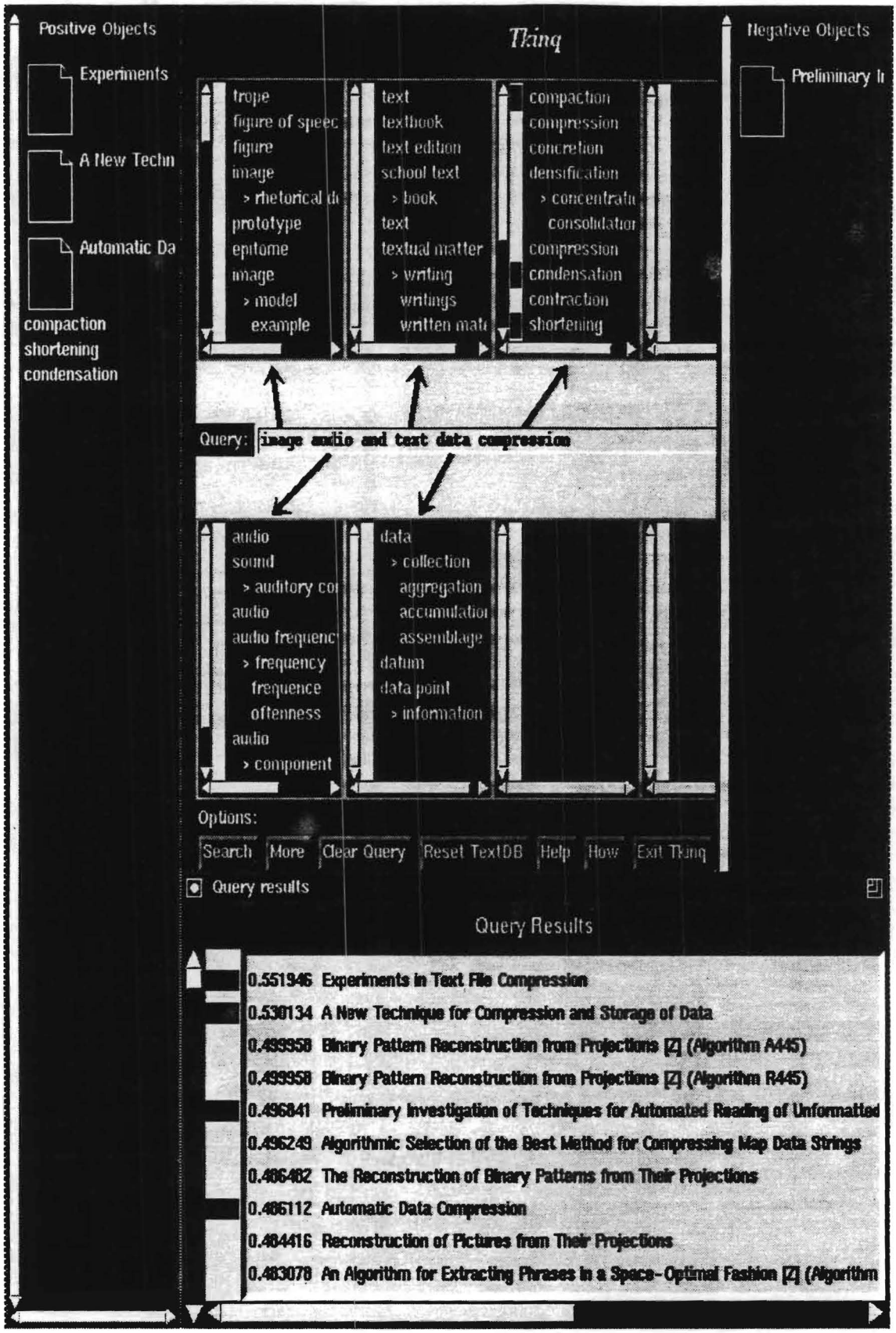


Figure 1: Sample querying session. The window titled "Positive object" is colored green and the window title "Negative Objects" is colored red. When a document is classified as positive/negative, the title of that document in the "Query results" window is also colored green/red.

file compression", "A new technique for compression and storage of data" and "Automatic data compression" as positive. The document titled "Preliminary investigation of techniques for automated reading of unformatted text" has been classified as negative. Internally, the systems extracts 4 - 6 high frequency words from the positive documents and adds it to the query thereby expanding the query. This results in the retrieval of documents similar to the positive documents.

- The user can also highlight a portion of a document and drag-and-drop it into the positive and negative windows. The words in the highlighted document portion are used to expand the query in the next iteration.

- During the next iteration, the reformulated query with the relevance feedback information is processed by the system resulting in an improved ranking of documents.

The positive and negative windows for feedback are aimed at mimicking the user's view that some information is in line with the information need and some not. After an object has been classified as positive (or negative), the system always colors the object green (or red) whenever the object is displayed, thereby reinforcing the user with the fact that the object is being used for relevance feedback. While arguing for the use of direct manipulation techniques for Information Retrieval, Mitev [Mit89] mentions that

> "Parts of document(s), individual word(s), sentences or groups of word(s) displayed could be used directly as something to be input for another search. This could be done, for example, by pointing and 'picking' them on the screen and carrying them across another area of the screen. The user would not have to input them again."

This is precisely what has been accomplished in our interface. In their retrieval system, Campbell [CS] uses a cut-and-paste mechanism for relevance feedback by letting the user add portions of retrieved documents back into the query window.

This section dealt with the interaction technique to let the user provide relevance feedback information to the system. The next section deals with visual feedback from the system on how the query results were computed.

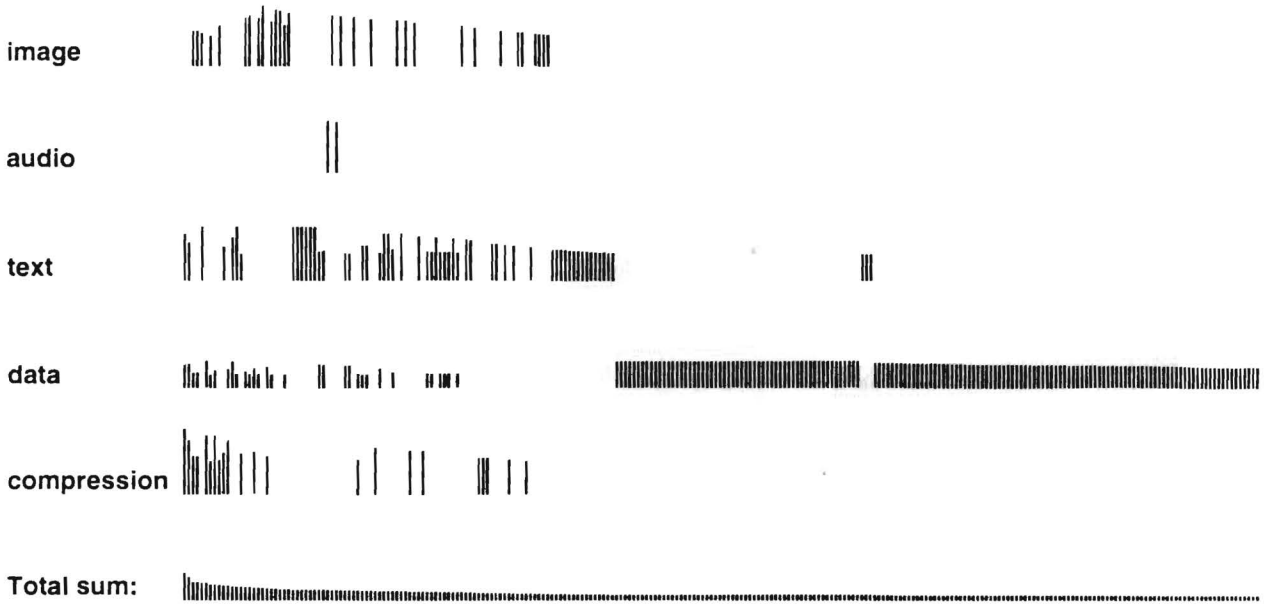


Figure 2: Visualization of results for the base query.

4 Visualization of query results

While systems with a boolean retrieval model retrieve an unordered set of documents in response to a query, ranked output information retrieval systems retrieve a ranked set of documents. While the reason for retrieving a document is fairly clear in the case of a boolean system, the reason why a document is assigned a specific rank is not apparent in the case of a ranked output system. Without knowing how the system computed the ranking of documents, the user will have to treat the retrieval mechanism as a black box. We stand to gain a lot by keeping the user more informed about the retrieval process of the system. If the user has more information about how the ranking was computed, he/she will be in a better position to reformulate the query for the next iteration. He/she can take into account the deficiencies of the system in adjusting his/her query. It will also help in reinforcing the right mental model.

In our interface, we keep the user informed about the retrieval mechanism by providing visual feedback about how the query results are related to the query words. This is done by a visualization scheme as shown in the figure 2. The visualization reveals the extent to which each query word was responsible for retrieving the set of documents. The visualization consists of a set of histograms, one for every query

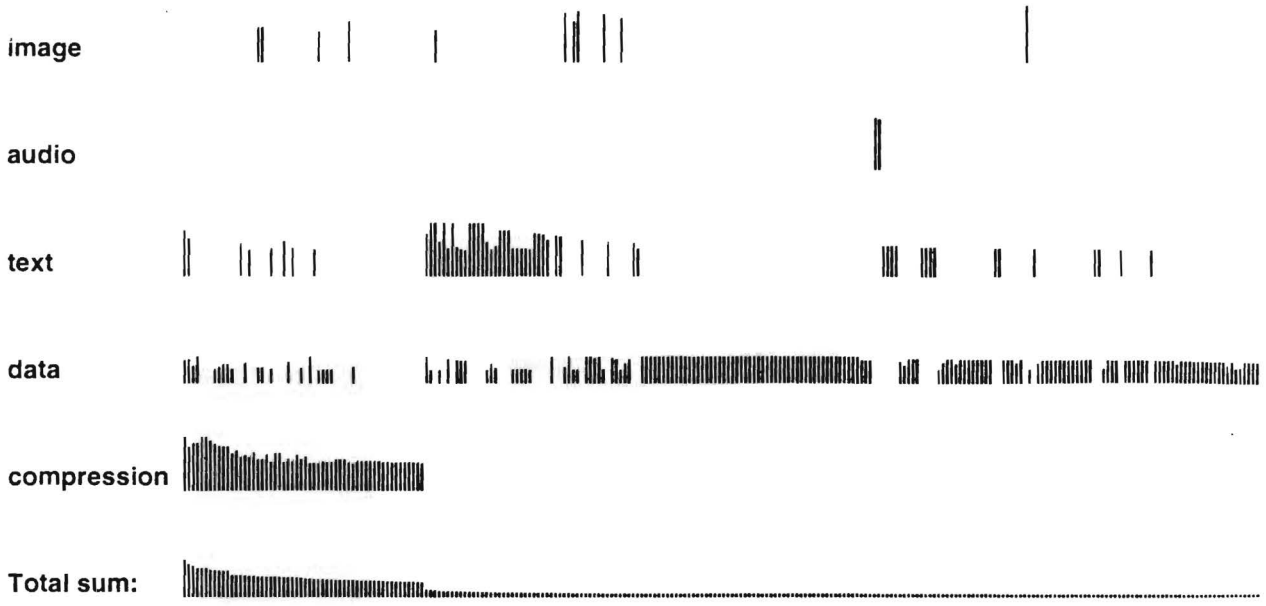


Figure 3: Visualization of results for query with feedback information.

word (except stop words) typed in by the user, and one histogram for the total query (labeled "Total sum"). All the histograms are placed one below the other with the "Total sum" histogram appearing at the bottom and the query-word-histograms appearing in the order in which query words were typed in. Each histogram consists of a set of vertical bars, one bar for each retrieved document. For the top ranked document, a vertical bar is drawn in the leftmost position (i.e, lowest X coordinate position) in the "Total sum" histogram. The height of the bar is proportional to the weight of the document. (Note that each document is given a weight. The higher the document weight, the more likely it is to be relevant to the query.) For the same document, vertical bars in the same X-coordinate position are also drawn in the query-word-histograms. The height of the vertical bar in any given query-word-histogram is proportional to the weight of the query word in that document. It represents the contribution of the query word in retrieving that document. If the query word does not appear in the document, thereby getting a weight of zero, a bar of zero height is drawn which shows up as an empty space in that X-coordinate position. The second ranked document occupies the next higher X-coordinate to the right and so on upto a maximum of top 200 documents.

The visualization shown in Figure 2 corresponds to the base query with no feedback information from the user. We can see that all but two of the top 200 documents

have nothing to do with audio. Almost all of the second half of the 200 documents were retrieved because they contained the query word "data". More significantly, only about 10% of the documents have anything to do with compression - which is the crux of the query. This illustrates that the query should be expanded with more words related to "compression". In fact, the decision to classify the three synonyms of "compression" (as shown in figure 1) was made after examining the distribution of "compression" in the visualization. Figure 3 shows the distribution of query terms in the query result for the revised query in the second iteration with all the feedback information. We can see that almost all the documents about "compression" have been ranked at the very top. Also there are more documents retrieved due to "compression" because of the synonyms and the positively classified documents. Our experience with this visualization scheme has shown it to be very useful in identifying differents facets of the query.

5 Conclusion & Future work

A prototype interface for a ranked output information retrieval system has been implemented. The interface facilitates the inherently interactive nature of the information seeking process. Drag-and-drop operations form the basis of interaction encouraging the user to provide feedback information to the system and helps in the dialog between the user and the system. Almost any information on the screen can be used by the user to provide feedback information. An online thesaurus, WordNet [Mil85], is integrated with the interface to form a single system.

The interface also supports a visualization scheme which illustrates how the query results are related to the query words. Visualizing the results of the query keeps the user more informed on how the system computed the ranking of documents. With this information, the user is better equipped to reformulate the query for the next iteration. It is our opinion that integrating all of the above features in a seamless interface leads to an interplay between different items that is much more beneficial than the sum of the individual items in isolation.

In demonstrating the system to the reference librarians at Georgia Tech and in observing casual users of the system, we believe that the features we have implemented in this system contributes to enhancing the end-user's interaction with the system. As a result, the system is better able to access the user's need and the user has a better understanding of the system's inference. However we cannot categorically conclude the effectiveness and the utility of the interface without conducting formal user-studies.

In future, we plan to test the effectiveness of the interface by conducting two studies: One with users having real information needs searching a traditional library database and another with volunteers searching the TREC [TRE94] document collection with supplied search statements. Since all the relevant documents for the supplied search statements in the TREC collection are known, recall and precision of searches performed with our interface can be compared against other systems.

6 Acknowledgments

We are thankful to Dr. Bruce Croft for letting us use the INQUERY retrieval system.

References

[Bat] Marcia J. Bates. Design for a subject search interface and online thesaurus for a very large records management database. In *Proceedings of the Annual Meeting of the American Society for Information Science*, pages 20–28.

[BC87] Nick Belkin and W.B. Croft. Retrieval techniques. In E. Martha, editor, *Annual Review of Information Science Technology*, pages 110–145. Elsevier Science Publishers, 1987.

[BMA+91] N.J. Belkin, P.G. Marchetti, M. Albrecht, L. Fusco, S. Skogvold, H. Stokke, and G. Troina. User interfaces for information systems. *Journal of Information Science*, 17:327–344, 1991.

[BMC93] N.J. Belkin, P.G. Marchetti, and C. Cool. Braque: Design of an interface to support user interaction in information retrieval. *Information Processing and Management*, 29(3):325–344, 1993.

[CCH92] J.P. Callan, W.B. Croft, and S.M. Harding. The inquery retrieval system. In *Third International Conference on Database and Expert Systems Applications*, September 1992.

[CRM91] S. Card, G Robertson, and J. Mackinlay. The information visualizer, an information workspace. In *Proceedings of CHI 91 Human Factors in Computer Systems.*, 1991.

[CS] I. Campbell and M. Sanderson. Personal communication. University of Glasgow.

[Dal90] P.W. Dalrymple. Retrieval by reformulation in two library catalogs: toward a cognitive model of searching behaviour. *Journal of the American Society for Information Science*, 41(4):272–281, 1990.

[FFS+93] Edward A. Fox, Robert K. France, Eskinder Sahle, Amjad Daoud, and Ben E. Cline. Development of a modern opac: From rectolc to marian. In Robert Khorfhage, Edie Rasmussen, and Peter Willett, editors, *Proceedings of sixteenth ACM SIGIR conference*, pages 248–259. ACM SIGIR, June-July 1993.

[Fid91a] Raya Fidel. Searcher's selection of search keys: I. the selection routine. *Journal of the American Society for Information Science*, 42(7):490–500, 1991.

[Fid91b] Raya Fidel. Searcher's selection of search keys: Ii. controlled vocabulary or free-text searching. *Journal of the American Society for Information Science*, 42(7):501–514, 1991.

[Fid91c] Raya Fidel. Searcher's selection of search keys: Iii. searching styles. *Journal of the American Society for Information Science*, 42(7):515–527, 1991.

[Har92] Donna Harman. User-friendly systems instead of user-friendly frontends. *Journal of American Society for Information Science*, 43(2):164–174, 1992.

[HB92] Micheline Hancock-Beaulieu. User friendliness and human-computer interaction in online library catalogues. *Program*, 26(1):29–37, January 1992.

[HB94] Scott Henninger and Nick Belkin. Tutorial on interface issues and interaction strategies for information retrieval systems. In *Human Factors in Computing Systems CHI 94 Conference Companion*, pages 387–388, 1994.

[HKW94] Matthias Hemmje, Clemens Kunkel, and Alexander Willet. Lyberworld – a visualization user interface supporting full text retrieval. In *Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, pages 249–259, 1994.

[Kor91] Robert Korfhage. To see, or not to see – is that the query? In *Proceedings of the 14th Annual International ACM/SIGIR conference on Research and Development in Information Retrieval*, pages 134–141, 1991.

[Mil85] G.A. Miller. Wordnet: A dictionary browser. In *Proceedings of the First Conference of the UW Centre for the New Oxford Dictionary.* Unversity of Waterloo, 1985.

[Mit89] Nathalie N. Mitev. Ease of interaction and retrieval in online catalogues: contributions of human-computer interaction research. In Charles R. Hildreth, editor, *The online catalogue*, chapter 8, pages 142–176. Library Association Publishing, London, 1989.

[Rij79] Keith Van Rijsbergen. *Information Retrieval.* Butterworths, London, second edition, 1979.

[SM83] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Company, New York, 1983.

[Spi93] Amanda Spink. Interaction with information retrieval systems: Reflections of feedback. In *Proceedings of the Annual Meeting of the American Society for Information Science*, pages 115–121, 1993.

[Spo94] Anslem Spoerri. Infocrystal: A visual tool for information retrieval and management. In *Human Factors in Computing Systems CHI 94 Conference Companion*, pages 11–12, 1994.

[SS92] Amanda Spink and Tefko Saracevic. Sources and use of search terms in online searching. In *Proceedings of the 55th Annual Meeting of the American Society for Information Science*, pages 249–255, 1992.

[TRE94] In D.K. Harman, editor, *The Second Text REtrieval Conference (TREC-2).* NIST Special Publication, March 1994.

[Wal87] Stephen Walker. Okapi: Evaluating and enhancing an experimental online catalog. *Library Trends*, Spring:631–645, 1987.

C-36-X21
#2

KNOWLEDGE BASED QUERY PROCESSING- (Project Summary-Tech Report)

by

Shamkant Navathe, P.I.
College of Computing
Georgia Institute of Technology
Atlanta, Georgia

July 1995

Submitted to Advanced Resaerch Project Agency
Grant No. F33615-93-1-1338

THE FOLLOWING REPORT WAS FILED WITH ARPA by email on July 5, 1995.

=== ADMINISTRATIVE DATA ===

1. ARPA ORDER NUMBER: A522

2. BAA NUMBER: BAA 92-06

3. CONTRACT/GRANT NUMBER: F33615-93-1-1338

4 AGENT: AIR FORCE - Wright Patterson Air Force Base, Ohio.

5. CONTRACT TITLE: Knowledge Based Query Processing
Known as the KBQP project

(official title: A Knowledge Based Approach to Integrating and Querying Distributed Heterogeneous Information Systems).

6. CONTRACTOR/ORGANIZATION: Georgia Institute of Technology

7. SUBCONTRACTORS: None

8. CO-PRINCIPAL INVESTIGATORS: Prof. Edward Omiecinski and Prof. Ashok Goel

9. CONTRACT:

9.1. ACTUAL START DATE: September 30, 1993
9.2. EXPECTED END DATE: January 31, 1997.

10. FUNDING PROFILE:

9.1. Current contract: $472,815.

9.2. Total funds provided to date for all years $388, 377

Total funds expended to date for all years $247 K

Report as of date May 31, 1995

9.3. Date total current funding will be expended: Dec. 31, 1995.

9.4. Funds required in FY96:
(i.e., funds needed to fund you from date in 9.3 above through 11/30/96) remaining balance of $84, 438.

===PRESENTATION CHARTS===

To be supplied later in the required format.

(SEE ATTACHED)

==========

http://www.cc.gatech.edu/computing/Database/database.html

The above page is being set up.

==========

10. ANYTHING ELSE YOU NEED (from ARPA):
Quick acquisition of software licensed under ARPA.

==========

===SIGNIFICANT EVENTS===

No events to report for FY 93.
==========

The following summary was posted according to the instructions given at the Web URL:

http://www.arpa.mil/sisto/project-sum.html
on July 5, 1995.

It is slightly modified and submitted below. Additionally, some viewgraphs are attached to give a complete idea of the overall thrust of the project.

=== PROJECT SUMMARY ===
(TECHNICAL REPORT)

ORGANIZATION: Georgia Institute of Technology

SUBCONTRACTORS: None

PRINCIPAL INVESTIGATORS:
Shamkant B. Navathe, Professor Phone: 404 894 0537
email:sham@cc.gatech.edu Fax 404 894 9442

CO- P.I.'s: Edward Omiecinski (edwardo@cc.gatech.edu)
Ashok Goel (goel@cc.gatech.edu)

TITLE OF EFFORT: "Knowledge Based Query Processing"

SUBTITLE: "HIPED: Heterogeneous Intelligent Processing for Engineering Design"

TEAM MEMBERS / GRADUATE STUDENTS:

Shamkant Navathe (PI)
Ashok Goel (Co-PI)
Edward Omiecinksi (Co-PI)

Jeffrey Donahoo (Graduate Student)
Yaakov Eisenberg (Graduate Student)
William Murdock (Graduate Student)
Jeffrey Pittges (Graduate Student)
Aravindam Veersamy (Graduate Student)
Sameer Mahajan (Graduate Student)

EXECUTIVE SUMMARY

The development of large-scale information/knowledge systems continues to be a cherished goal for computer/information science. Some of the difficulties arise due to the heterogeneity of information (i.e.,

content, structure, form), while others arise because of the heterogeneity of reasoning (e.g., rule-based reasoning, case-based reasoning, model-based reasoning). We are exploring several critical elements in the design of large-scale systems including query formulation, query optimization, knowledge compilation in query answering, the integration of multiple information sources, the integration of multiple methods of reasoning, and device ontologies for engineering design problems. Basic research on query formulation and semantic query optimization has resulted in one completed and one ongoing Ph.D. dissertation.

OBJECTIVE:
To address various aspects of query processing in large scale distributed heterogeneous environments with special emphasis on incorporating knowledge (meta-data) at different levels. The knowledge relates to information about the sources of data, their structure, their content, and their overall relevance to the problem at hand. Another objective is to incorporate knowledge compilation, schema integration and data integration in a flexible way. Finally, the overall gosl is to develop a prototype system that instantiates, evaluates and demonstrates the intelligent integration of concepts for the application domain of engineering design.

APPROACH:

Our present approach to the issue of incorporating knowledge into query processing and formulation can be broken down into six areas which can be grouped into two categories. The first one deals with query formulation, optimization, and answering. The second deals with knowledge and data integration.:

A. Query Formulation, optimization, and answering:

1. Query Formulation:

Query formulation using free form textual queries is investigated. The target databases contain of semi-structured data like text documents. User interface and visualization techniques are developed and tested to let the user discover the ways in which data is organized. A thesaurus is incorporated to guide the user. Current work addresses extensive studies with the user interface and visualization to determine the overall utility of the approach.

2. Semantic and Multiple Query Optimization:

Query optimization is a decision process that selects the best query evaluation strategy from a set of execution plans. The performance of this process can be improved by providing better information about the contents of the database (i.e., meta-data). A new approach called Meta-Data View Graphs(MVG) and their efficient maintenance in light of database updates has been developed.

3. Knowledge Compilation in Query Answering: In determining how to efficiently answer a query, one can learn from the history of the performance of previously dealt with queries. A solution technique using case-based reasoning is being developed where retrievals are compiled into meta-cases consisting of (query, answer, trace) triplets.

B. Knowledge and Data Integration

4. Integration of Knowledge Systems with External Information Sources: An existing autonomous knowledge system called Kritik that combines case-based and model-based reasoning for designing engineering devices

was developed earlier at Georgia Tech. It is being enhanced to link with existing databases of past design cases and components. It generates requests for data that are compiled and processed against the avilable databases.

5. Integration of Data from Multiple Information Sources: This is a classical problem of having to deal with data with heterogeneous models and systems. With our long-standing track record of work in the schema integration area, a practical approach to integration with partially integrated schemas based on meta-data is being developed.

6. Transparency of Knowledge-Based Reasoning: Knowledge organization and information processing in the integrated information source environments must be transparent to users. Three issues are under investigation: how to explain and illustrate the system's reasoning, how to explain and justify its results, and how to enable the user to navigate and browse its knowledge bases.

PROGRESS:

In the second year of this effort so far we have made considerable progress in each of the six areas listed above. Work in areas one and two is nearly complete and has resulted in two Ph.D. dissertations. Small prototype solutions have been developed in the remaining areas. The third year will involve further research in areas 3 through 5 above and will result in a prototype that underscores the theme: the use of meta-data in developing large scale knowledge based systems.

FY 95 ACCOMPLISHMENTS

A. Query Formulation, optimization, and answering:

In the area of query formulation, a prototype query formulation system has been implemented for dealing with real-life text databases with visualization and user feedback techniques. The prototype has been extended to work with an online library catalog and with a standard collection of full text documents, called the TREC collection (sponsored by NIST) containing TIPSTER databases. The effectiveness of the visualization scheme is being tested by conducting user studies. The studies are aimed at discovering:

a. the different ways in which end-users utilize the visualization and
b. the improvement in performance in terms of "recall" and "precision", the two standard measures used in the Information Retrieval community.

The MVG (Meta-data View Graph) is a metadatabase capable of maintaining the semantic and structural meta-data for views of a database. As the database evolves, this meta-data may need to be updated to reflect the current semantic content of the database. In the last year, work has concentrated on developing solutions to this approach based on update logs. Efficient schemes of updating the meta-data and corresponding query execution plans have been developed.

Work has begun in the third area of knowledge compilation. In knowledge compilation, various types of knowledge is used to enhance query processing: the meta-model, meta-associations and meta-cases. The meta-model can be thought of as the global schema, which can be constructed from the meta-data. Meta-associations are compiled from past experiences (i.e., from searching the meta-model). Meta-cases consist of the past experiences (the search plan followed) together with the query and its answer. Use of a deductive database system called CORAL is being considered for storing meta-data.

B. Knowledge and Data Integration

To investigate integration of knowledge systems with external information sources, knowledge of engineering components has been ported from Interactive Kritik to a relational database implemented in Oracle. Then LOOM and IDI were used to enable Interactive Kritik to access the Oracle database. In the area of data integration, we have developed a mechanism where administrators can describe their data. This description is then used to create a partially integrated representation of the data from a collection of information sources. We have started development of a tool to take the information source descriptions and produce an integrated data representation. In the area of transparency of knowledge-based reasoning, a graphical interface has been built to Kritik and used to partially explain and illustrate the system's reasoning. Tools for enabling different kinds of user interaction with the system such as enabling a user to explore, navigate and browse through the design cases in the systems knowledge base are under development.

PRODUCTS: NONE.

PUBLICATIONS: A list of publicly accessible papers published in technical journals, conference proceedings, magazines, etc. Give full citation as in a reference list for a technical publication.

1. S. B. Navathe and M.J. Donahoo, " Towards Intelligent Integration of Heterogeneous Information Systems," Sixth International Hong Kong Computer Society Database Workshop on Database Reengineering and Interoperability, March 3-4, 1995.

2. A, Veerasamy, S. Hudson and S. B. Navathe, " Visual Interfaces for Text Information Retrieval Systems," Proc. of Visual Database Systems -3, an IFIP WG 2.6 Workshop, Lausanne, Switzerland, March 1995,Chapman Hall Publishing.

3. J. Pittges, Improving Query Optimization with Instance Based Constraints, Proc. IFIP WG 2.6 Working Conference on Database Semantics (DS-6) , Stone Mountain, GA, June 1995, Chapman Hall Publishing.

DATE PREPARED: July 5, 1995

SIGNIFICANT EVENTS:

1. A user interface tool has been developed to work against large document databases. It incorporates the INQUERY information retrieval engine combined with the WORDNET Thesaurus. Currently, the tool has been integrated with the TIPSTER data and is undergoing user field testing. We will also be competing in the TREC conference to compare our system with other information retrieval systems worldwide.

2. A Ph.D. dissertation on semantic query optimization by Jeff Pittges has been completed.

FY 96 PLANS

1. User studies on the query formulation interface will be completed by September 1995. Based on the results of the user study, we plan to go through an iterative design cycle of the interface to better suit the end-user needs. The study will also shed light on which kinds of

visualization suits the information searching process.

2. Current set of integration tools will be enhanced to accomplish large-scale integration. This incorporates:

a). Construction of Augmented Export Schema development tools to allow information source administrators to completely/accurately represent their data.

b). Extend the reasoning capabilities of schema integration tool by allowing more types of inferencing on schema correspondences and consideration of user feedback.

3. Development of query processing facilities which support users searching a very large information space. Such facilities include explanation of query results, suggestion of future query directions, and visualization of information relevancy. We will use a deductive database system (DDBS) to store and access the meta-data. The main advantage is that the DDBS will provide us with a built-in inferencing capability. The system will allow us to make inferences over IS-A, PART-OF and more general relations which we define in the DDBS. Also, from searching the meta-data, the query processor will determine the different information sources that will be searched to process the query. To aid in processing the same or similar queries, we will use knowledge compilation techniques.

TECHNOLOGY TRANSITION:

We are presently investigating the potential use of our prototype system for the design of demanufacture processes such as product service and disassembly processes. This work is in collaboration with a group of mechanical engineers at Georgia Tech.

We are also exploring the potential use of our system for the procurement of engineering components, devices and systems. This work is in collaboration with a group of computer and logistics scientists at Michigan State University.

In addition, in collaboration with Warner Robbins AFB and the Air Force, we are exploring the
potential use of our system for accessing information about Air Force Alternative Fuel Vehicle (AFV) Program.

QUAD Charts

Background:

Sham Navathe has spent many years working in the area of information integration (e.g. CANDIDE Project, etc). The technologies developed in those projects and the recognition of the need for large-scale integration of information sources led to the proposed research direction of this project. Ashok Goel has spent many years working in the area of knowledge representation and reasoning (e.g. Kritik, Ideal, etc). We hope the reasoning/representation techniques from his work can be applied to query processing. In addition, large-scale knowledge bases resulting from the integration work in our project will facilitate scaling of the reasoning systems developed by Ashok Goel.

Technical Approach:

1. Develop a method to represent information sources. Then, create a procedure to take information source specifications and produce a global representation of all the data in the information sources.

2. Develop techniques to efficiently query the data from the information sources. Semantic query optimization, query result visualization, and case/model-based reasoning will be used to augment the functionality of traditional database query systems.

3. Test the usability and scalability of the knowledge of the system by allowing an intelligent system to attempt to utilize the information available after integration. Kritik will be used to test if the HIPED system provides the necessary data for its reasoning processes.

Objectives:

1. Large-scale integration of information sources: This will provide DoD with the capability to perform decentralized integration of a large number of databases.

2. Access of a large collection of data: This allows access of a large amount of loosely related data. Traditional database query processing is only suited for domain specific searching with a relatively small schema.

3. Integration of intelligent systems with large, non-application specific knowledge bases: This will allow intelligent systems to utilize knowledge which is useful in their reasoning processes that is widely distributed and heterogeneous.

Technical Challenge:

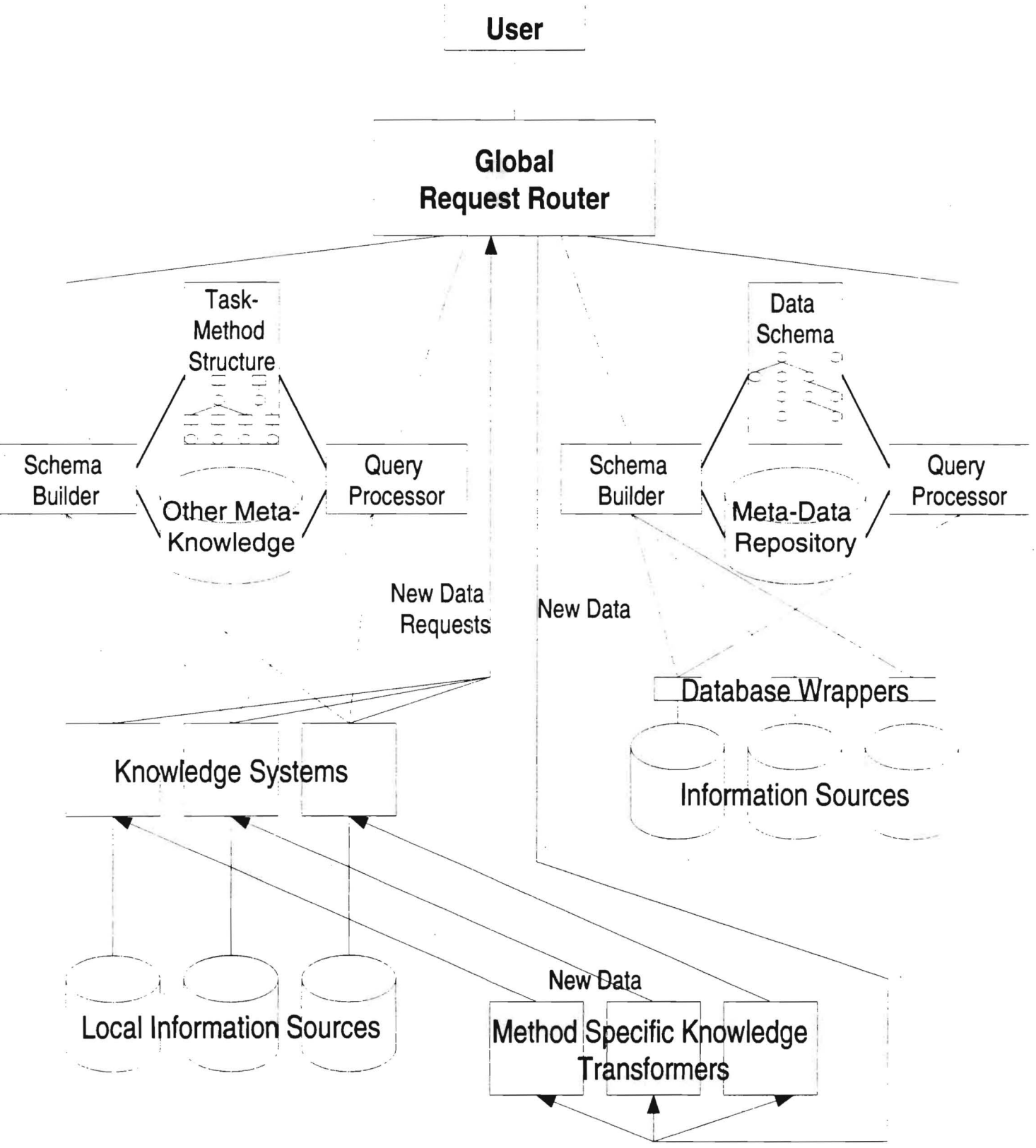
1. One of the main difficulties is defining a method to properly describe the semantics of an information source. We will use basic knowledge representation constructs (e.g. associations, domain concept hierarchies, etc) to allow the administrator to relate their information source entities to other information source entities.

2. Due to the complexity of information source semantics, it is intractable to automatically generate a completely integrated representation of all the data. We will be using various reasoning techniques (case/model-based) and meta-data to aid a user in finding information which will be difficult to find since the integration is incomplete.

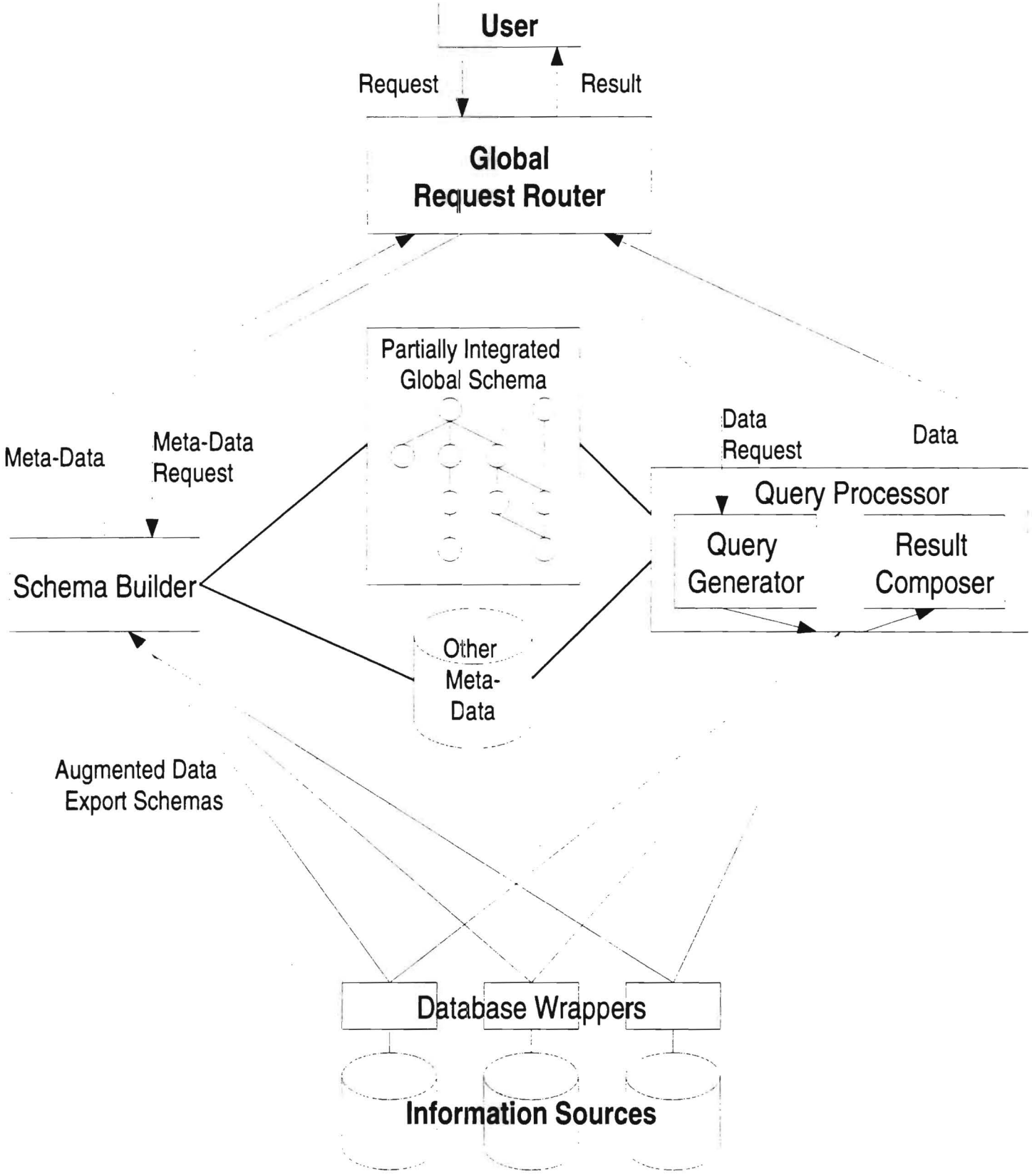
3. Knowledge is the key to successful reasoning in intelligent systems; however, ad hoc access of knowledge from other systems is difficult. The semantic description of each information source in concert with the query processor reasoning facilities will provide the necessary information to import knowledge for external bases into an intelligent system.

==========

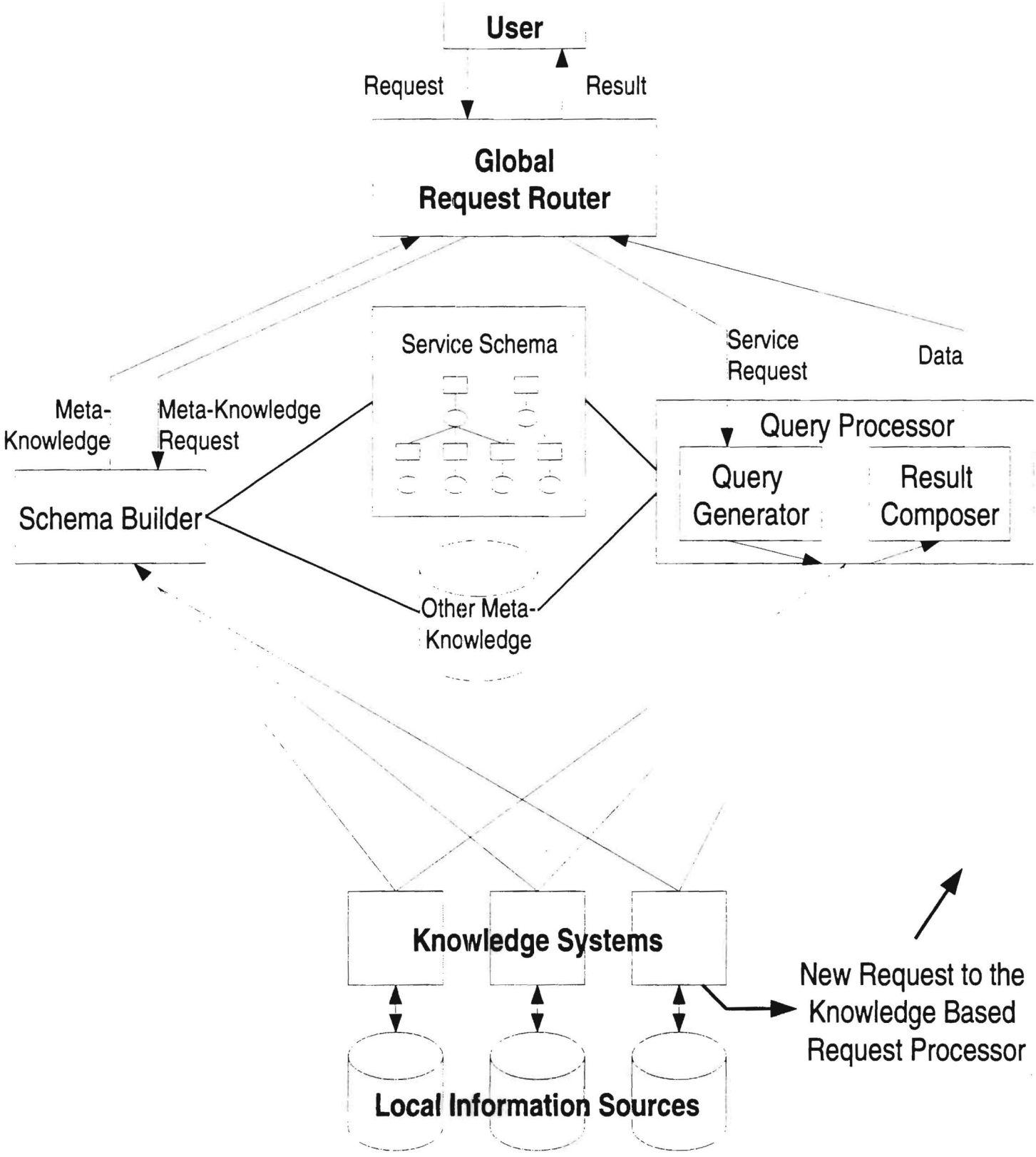
Integration of Information from DB & KB Systems



Research Approach 1:
Integration of Data from Database Systems

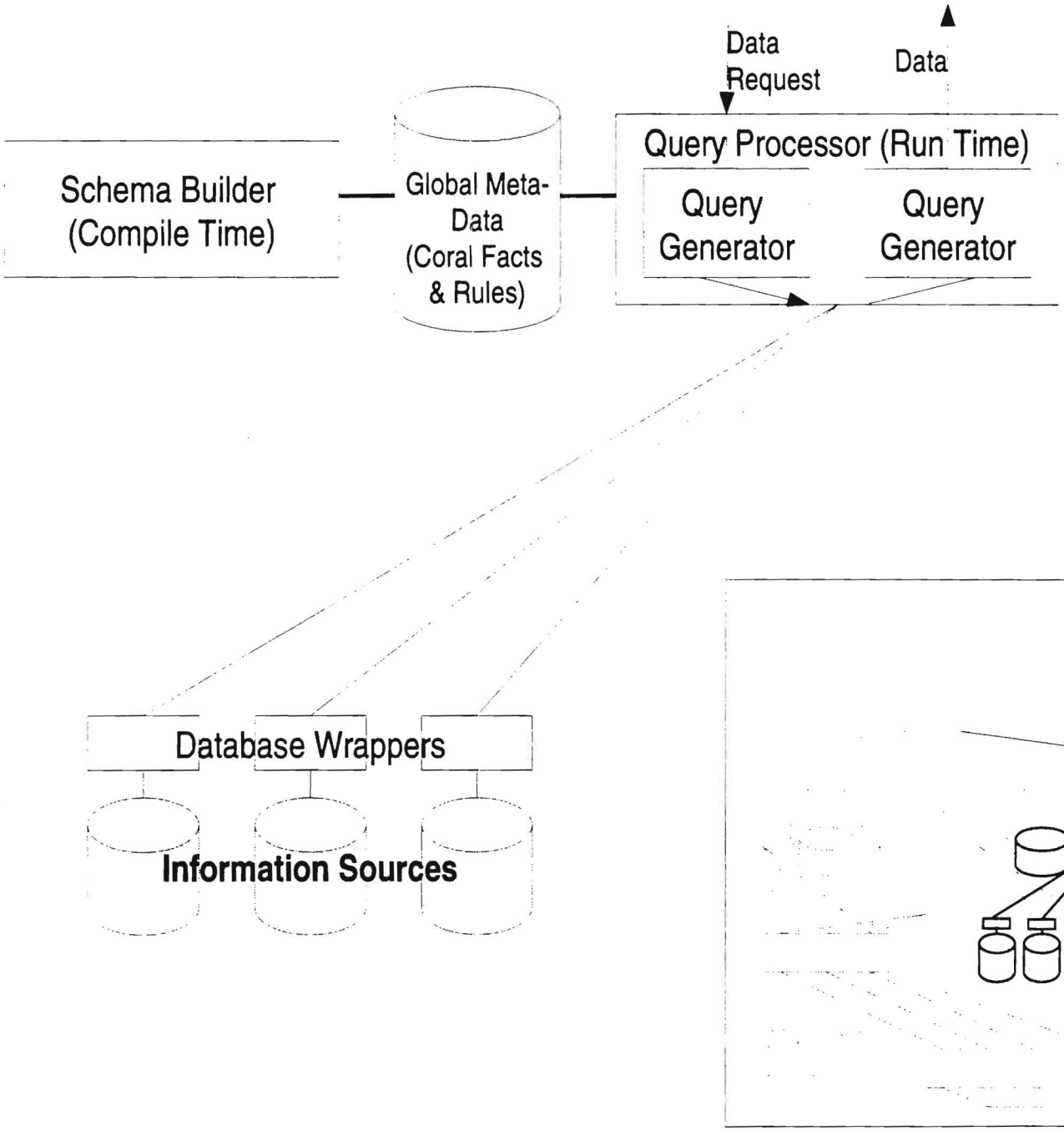


Reasearch Approach 2:
Integration of Services from Knowledge Systems



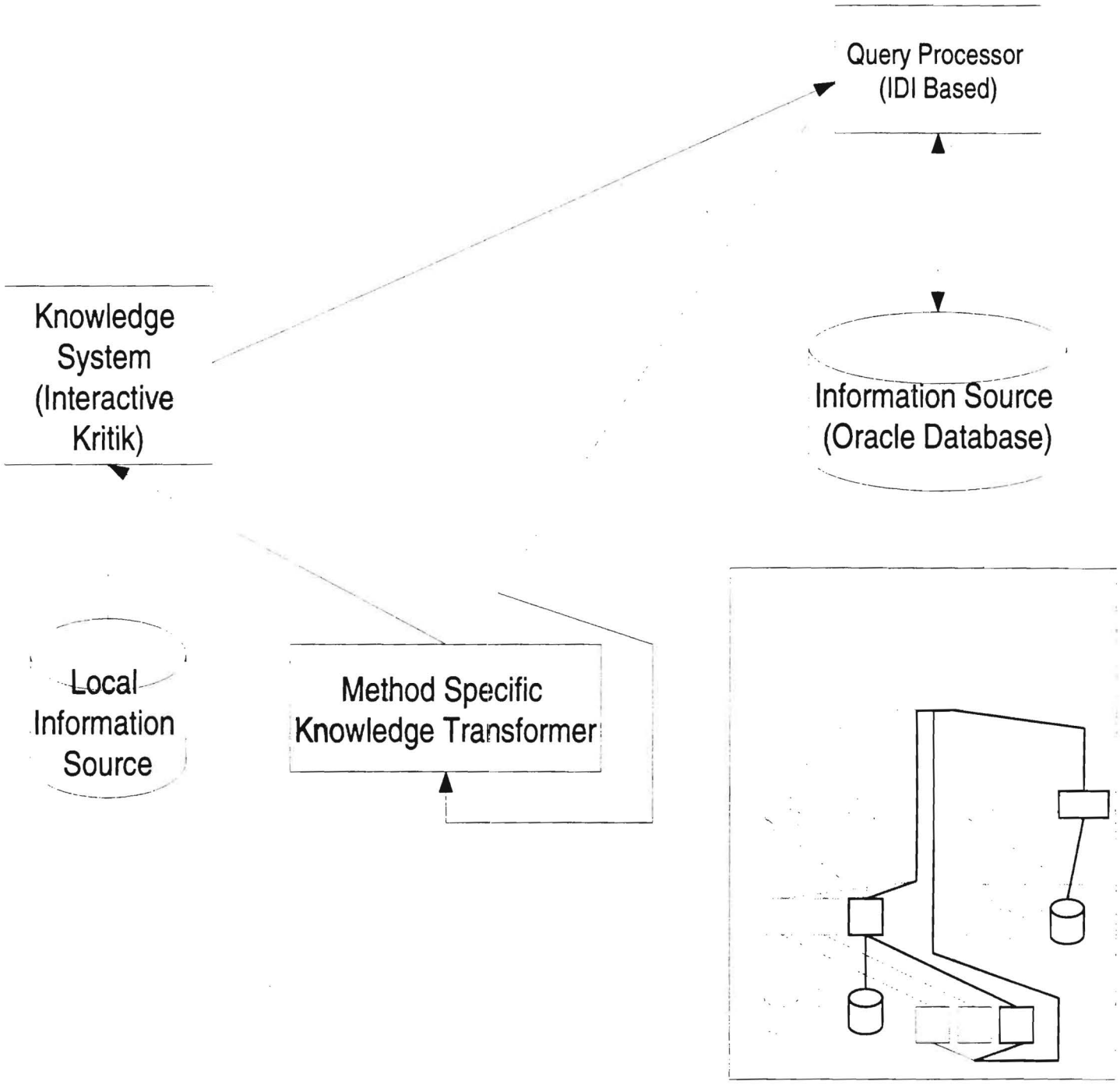
Current Implementation - 1

Large-Scale Information Source Integration
Rule Based Flexible Integration of Information Sources
Using the Coral System



Current Implementation - 2

Method Specific Knowledge Transformation
Connection of Knowledge System with External Database



C.36-
6

Georgia Tech

College of Computing

Georgia Institute of Technology
Atlanta, Georgia 30332-0280
USA
404•894•3152
404•853•9378 FAX

May 2, 1997

Charles Satterthwaite
WL/AASH Bldg. 620
2241 Avionics Circle
Wright-Patterson AFB
Ohio, 45433-7318

Re: Contract No. F 33615-93-1-1338 - Annual and Final Report for "KBQP: A Knowledge-Based Approach to Integrating and Querying Distributed Heterogeneous Databases".

Dear Mr. Sattherthwaite:

Enclosed please find a complete set of publications that we produced on our project entitled: "KBQP - A Knowledge-Based Approach to Integrating and Querying Distributed Heterogen Databases".

Please note that I am sending you separately by e-mail the material that will go as the front matter explaining the contents of these papers. Once you have marked up the other material, I will send you the final formatted copy so you can bind and circulate copies. Thank you for your cooperation.

Sincerely,

Shamkant B. Navathe
Principal Investigator

SBN/gb

cc: Anita Rowland, Office of Contract Administration ✓

Enclosures

PART 1

HIPED: Heterogeneous Intelligent Processing for Engineering Design

PUBLICATIONS (PART1):

1." Towards Intelligent Integration of Heterogeneous Information Sources ," Shamkant B. Navathe and Michael J. Donahoo.. In *Proceedings of the 6th International Workshop on Database Re-engineering and Interoperability*, Computer Society of Hong Kong, March 1995.

2. " Rule Based Database Integration in HIPED : Heterogeneous Intelligent Processing in Engineering Design ", Shamkant B. Navathe, Sameer Mahajan, Edward Omiecinski . In *Proceedings of International Symposium on Cooperative Database Systems for Advanced Applications*, World Scientific Press, 1996.

3. "Integrating Heterogeneous Databases for Engineering Design," Sameer Mahajan and Shamkant B. Navathe, submitted to Conference on Deductive Databases and Logic Programming, Leuven, Begium,1997.

4. " From Data to Knowledge: Method-Specific Transformations," Michael J. Donahoo, J. William Murdock, Ashok K. Goel, Shamkant B. Navathe, Edward Omeicinski, submitted for publication.

PART 2

VISUALIZATION AND USER INTERFACE TECHNIQUES FOR INFORMATION RETRIEVAL

PUBLICATIONS (PART2):

1. Visual Interface for Textual Information Retrieval Systems", Aravindan Veerasamy, Scott Hudson, Shamkant Navathe. In *Proceedings of IFIP 2.6 3rd Working Conference on Visual Database Systems* 1995, Elsevier, North Holland, pp. 333-345.

2. ``Querying, Navigating and Visualizing a Digital Library Catalog", Aravindan Veerasamy, Shamkant Navathe. In *Second International Conference on the Theory and Practice of Digital Libraries* ,June 11-13, 1995, Austin, TX

3.``Interactive TREC-4 at Georgia Tech", Aravindan Veerasamy. In *Fourth Text REtrieval Conference*, Oct, 1995, Gaithersberg, MD

4. ``Evaluation of a tool for visualization of information retrieval results", Aravindan Veerasamy, Nick Belkin. In *Proceedings of the SIGIR 1996, the 19th Annual International Conference on Research and Development in Information Retrieval.*, ACM, New York.

5. "Effectiveness of a graphical display of retrieval results", Aravindan Veerasamy and Russell Heikes. In *Proceedings of the SIGIR 1996, the 19th Annual International Conference on Research and Development in Information Retrieval.*. ACM, New York.

PART 3

Metadata Management for Intelligent Query Processing

PUBLICATIONS (PART3):

1. Jeff Pittges. " Maintaining Instance-Based Constraints for Semantic Query Optimization," In *Proceedings of the Sixth IFIP TC-2 Working Conference on Data Semantics (DS-6)* , Stone Mountain, Georgia, May1995

2. " Maintaining Semantic and Structural Metadata in the Metadata View Graph," J. Pittges, L. Mark, and S. Navathe. In *Proceedings of the Seventh International Conference On Management of Data*, Pune, India, December 1995.

C-36-X21
#3, 4, 5, 6

DEPARTMENT OF THE AIR FORCE

WRIGHT LABORATORY (AFMC)
WRIGHT-PATTERSON AIR FORCE BASE, OHIO

19 June 1997

FROM: WL/AASH, Bldg 635
2185 Avionics Circle
Wright-Patterson AFB OH 45433-7301

SUBJ: Receipt of Technical Papers and Draft Write-up towards Final Report for Grant F33615-93-1-1338 entitled "A Knowledge-Based Approach to Integrating and Querying Distributed Heterogeneous Information (also called HIPED"

TO: Georgia Institute of Technology College of Computing
ATTN: Dr. Sham Navathe

1. Wright Laboratory has received 11 Published Technical Papers (attached with this letter) and a program synopsis relating these papers to the HIPED program, a Defense Advanced Research Projects Agency (DARPA) Intelligent Integration of Information (I3) Broad Area Announcement (BAA) effort. These papers an synopsis will serve as the Final Report for this effort.

2. Wright Laboratory is currently reviewing these submitted materials for publication into its Defense Technology Information Center (DTIC) and expects this to be finalized in the later part of July.

3. The Project Engineer for this effort is fully satisfied with the work performed under this Grant, and with the written materials received to date. He is waiting for the DD 250, so that final payments can be made to Georgia Tech.

4. Please contact Mr. Charles Satterthwaite (WL/AAAF-2) if you require further information at DSN 785-3947 or commercial 513-255-3947.

CHARLES P. SATTERTHWAITE, Project Engineer
WL/AASH, 937-255-6548 ext. 3584

FINAL REPORT

A Knowledge-Based Approach to Integrating and Querying Distributed Heterogeneous Information Systems

(Also called HIPED: Heterogeneous Intelligent Processing for Engineering Design)

SPONSOR: DARPA (Defense Advanced Research Project Agency) under the I3 (Intelligent Integration of Information) program.

submitted to

Department of the Air Force
Wright Laboratories
Wright Patterson Air Force Base, Ohio

under contract number F 33615-93-1-1338

by

Prof. Shamkant B. Navathe, P.I.
Colege of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

PART 1

HIPED: Heterogeneous Intelligent Processing for Engineering Design

PUBLICATIONS (PART1):

[1.1]." Towards Intelligent Integration of Heterogeneous Information Sources ," Shamkant B. Navathe and Michael J. Donahoo.. In *Proceedings of the 6th International Workshop on Database Re-engineering and Interoperability*, Computer Society of Hong Kong, March 1995.

[1.2]. " Rule Based Database Integration in HIPED : Heterogeneous Intelligent Processing in Engineering Design ", Shamkant B. Navathe, Sameer Mahajan, Edward Omiecinski . In *Proceedings of International Symposium on Cooperative Database Systems for Advanced Applications*, World Scientific Press, 1996.

[1.3]" From Data to Knowledge: Method-Specific Transformations," Michael J. Donahoo, J. William Murdock, Ashok K. Goel, Shamkant B. Navathe, Edward Omeicinski, to appear in *Proceedings of the International Conference on Methodologies for Intelligent Systems* (Z. Ras, Ed.), IEEE Press, 1997.

[1.4] "Integrating Heterogeneous Databases for Engineering Design," Sameer Mahajan and Shamkant B. Navathe, Working Paper.

Towards Intelligent Integration of Heterogeneous Information Sources*

Shamkant B. Navathe Michael J. Donahoo

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
{sham, mjd} @ cc.gatech.edu

Abstract

Current methodologies for information integration are inadequate for solving the problem of integration of large scale, distributed information sources (e.g. databases, free-form text, simulation etc). The existing approaches are either too restrictive and complicated as in the "federated" (global model) approach or do not provide the necessary functionality as in the "multidatabase" approach. We propose a hybrid approach combining the advantages of both the federated and multidatabase techniques which we believe provides the most feasible avenue for large scale integration. Under our architecture, the individual data site administrators provide an *augmented export schema* specifying knowledge about the sources of data (where data exists), their structure (underlying data model or file structure), their content (what data exists), and their relationships (how the data relates to other information in its domain). The augmented export schema from each information source provides an intelligent agent, called the "mediator," knowledge which can be used to infer information on some of the existing inter-system relationships. This knowledge can then be used to generate a partially integrated, global view of the data.

1 Introduction

Much of the research in database interoperability has focused on two extremes: multidatabase and federated systems. Multidatabase [Lit90, Spe88] systems provide a uniform access language to a set of database systems. While this is a necessary first step in solving the problems of heterogeneity, it places most of the integration responsibility on the user which may be unacceptable. Federated systems[She90] propose to create a global view of the underlying systems making the heterogeneity completely transparent to the user. While this approach is enticing, the complexity of constructing a global schema for large scale integration makes this approach infeasible because it requires an administrator who understands the semantics of all underlying systems and can resolve all inter-system schematic conflicts[Bat86]. In addition, the maintenance of a global schema in the face of addition/deletion of systems is difficult.

A better approach to interoperability involves the combination of techniques of reasoning and learning with techniques of data modeling and access to provide a partially integrated, global view. To accomplish this, the administrator of each underlying system presents a semantic description (augmented export schema) of their information to the "mediator." This augmented export schema may be as simple as the typical export schema or as detailed as a knowledge-based data description of the data, its relationships, and the system's domain. A knowledge-base system, such as Loom[Bri94], provides the capability to represent knowledge about the underlying information repositories and to make inferences as to the relationships among the various autonomous systems and generalizations concerning the information in each system. We have previously demonstrated that classification hierarchies can be effectively used to carry out integration of schemas[Sav91]. In this paper, we

*To appear in Proceedings of 6th International Hong Kong Computer Society Database Workshop, Hong Kong, February 1995

review the goals and strategy of the project HIPED. Heterogeneous Information Processing for Engineering Design, which we are currently pursuing at the Georgia Institute of Technology.

2 Related Work

Earlier work in integration provides the motivation and framework for our efforts. Batini et al. [Bat86] detail the problems of schema integration and provide a methodology for comparison of proposed solutions. Unlike many earlier integration efforts, we do not limit ourselves strictly to integration of databases. Instead, we focus on the integration of *information sources* including databases, free-form text, hypertext, etc. One possible method of dealing with this wide variety of information is to use Stanford's Object Exchange Model (OEM)[Pap94] which allows information exchange via *self-described* objects[Mar85] between different types of information sources. We propose to adapt the mediator paradigm[Pap94, Wei92, Wei93, Are94] to perform integration of the augmented export schemas. Integration of heterogeneous information sources requires a semantically rich data model. Earlier work has shown that the CANDIDE[Bec89, Nav91] model provides unique integration capabilities not found in traditional models. One major feature of the CANDIDE model is its ability to compute class-subclass relationships even among classes from dissimilar systems by subsumption from class relationship information[Sav91, She93, Wha93, Bra85]. Work with classification in the object-oriented model has produced similar results[Nav95, Are]. A variety of such systems supporting description logics are surveyed in [Bor94].

3 Approach

Our main objective is to build and demonstrate an intelligent interface to a set of (possibly autonomous) information sources including structured databases, knowledge bases, and unstructured data. Figure 1 shows our proposed architecture. The parenthetical references are made to applications developed under the ARPA I3 Initiative. KQML (Knowledge Query and Manipulation Language)[Cha92] allows remote access to knowledge/data bases. LIM (Loom Interface Module)[Par93b] allows import of external database information into Loom data structures. IDI (Intelligent Database Interface)[Par93a] is a common access language to several commercial database systems.

The approach we have selected involves development of an Engineering Design Mediator (EDM) which utilizes meta-knowledge of the underlying information to aid a user in "browsing" the data for relevant information sources and to make informed decisions about a plan for retrieving the appropriate data. To demonstrate this technology, we intend to augment the capabilities of both an autonomous (KRITIK2) and an interactive (Canah-Chab[Goe93]) device design system by providing a mediated interface between the design system and a collection of data/knowledge based systems (D/KBS). The mediator will be responsible for processing queries from the device design systems by determining where relevant data is, sending the appropriate query to the information site, performing the appropriate translations on the data, and returning the data to the design system. The design of the mediator is predicated on the following design goals:

1. Autonomy of the remote systems. Additionally, the remote systems should not be required to perform any functions outside of those defined for the internetwork connecting the system to the mediator.

2. Meta-data query facilities which allow the design system to determine relevant information about component parameters, previous design specifications, device function descriptions, etc. The mediator may also take an active role in helping the design tool determine what information may be helpful (e.g. by use of a thesaurus, domain concept hierarchy, etc).

3. Separation of concerns of the device design system from the query system. This will facilitate reuse of the mediated query system for other intelligent tasks such as planning.

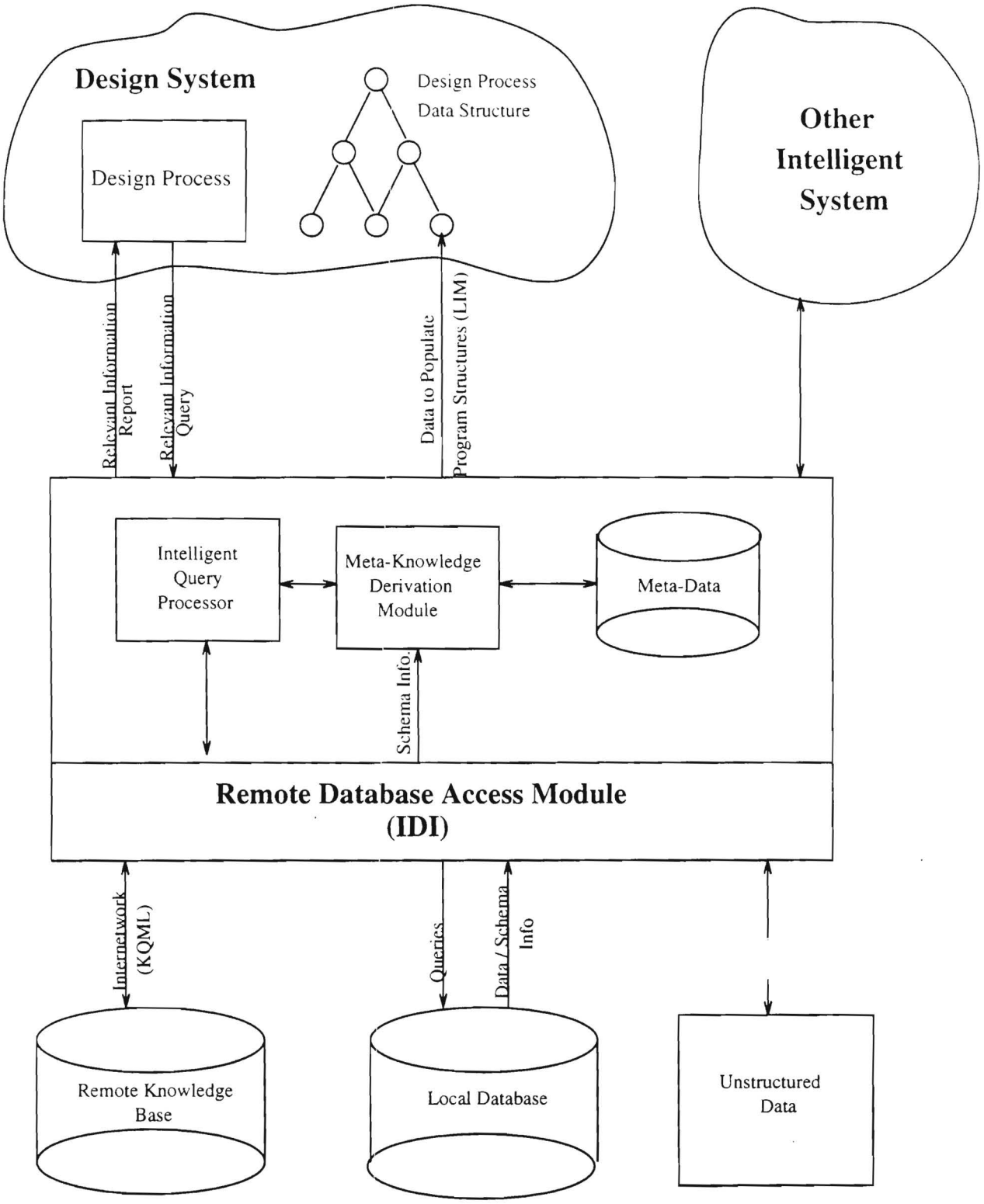


Fig. 1: Proposed Architecture for the Engineering Design Mediator (EDM)

4. Data location (remote vs. local) and data organization (relational, knowledge base, text, etc) transparency.

5. Easy import of external D/KBS information into existing design system data structures minimizing the required changes to the device design system.

These constraints are designed to facilitate reuse of the mediator and to make the use of the system as transparent to intelligent applications as possible. Figure 2 presents an example query processing scenario.

4 Ongoing Research

Research is currently under way in the following areas to facilitate construction of a prototype query system which can be integrated with the device design system:

- Selection and development of the appropriate export data model to represent the data stored at each information source.
- Construction of an export knowledge model whereby information source administrators can express the relationships between their data and real world domain concepts. This in combination with the export data model will define the *augmented export schema.*
- Development of techniques for providing integration of the schemas of information sources into a partially integrated, global schema.
- Determination of optimization techniques for querying the remote information sources. Since the information sources may be interconnected with a WAN, a query processing bottleneck may arise with frequent remote data transmission.
- Provision of a query interface which aids the user in deriving the best answer to a query. Since no completely integrated schema exists and the user does not know what information is available, a query processor is required to guide users to the desired information.
- Capability of inferencing intersource knowledge from the augmented export schemas specifically concerning the relationships between information source entities.
- Ability to learn new, relevant knowledge about information sources based on user interaction.

5 Future Direction

Our initial focus is on providing access of integrated information to intelligent device design systems, but many other applications of this technology exist. With the advent of internetworks which connect thousands of computers all over the world, an explosion has resulted of the available data, both unstructured (text, graphical documents, audio, video, program sources) and structured (under DBMS control), accessible to hundreds of thousands of users. It would be difficult, if not impossible, to integrate all these sites with the current heterogeneous database techniques especially since most sites will not all be willing to provide services beyond those defined by the internetwork. Many query applications already exist for the Internet. WAIS servers provide keyword access to documents; however these documents must be under the control of a WAIS server. Gopher allows sites to setup directories of information that users can browse, but the information can only be accessed in the organization defined by the site manager. Archie provides a keyword query interface to find source code, but the keywords only work on the name of the source file (the user cannot ask for a program that performs some function, X; instead they must find the name of a program that performs X and search for it by name. World Wide Web (WWW) provides a nice interface

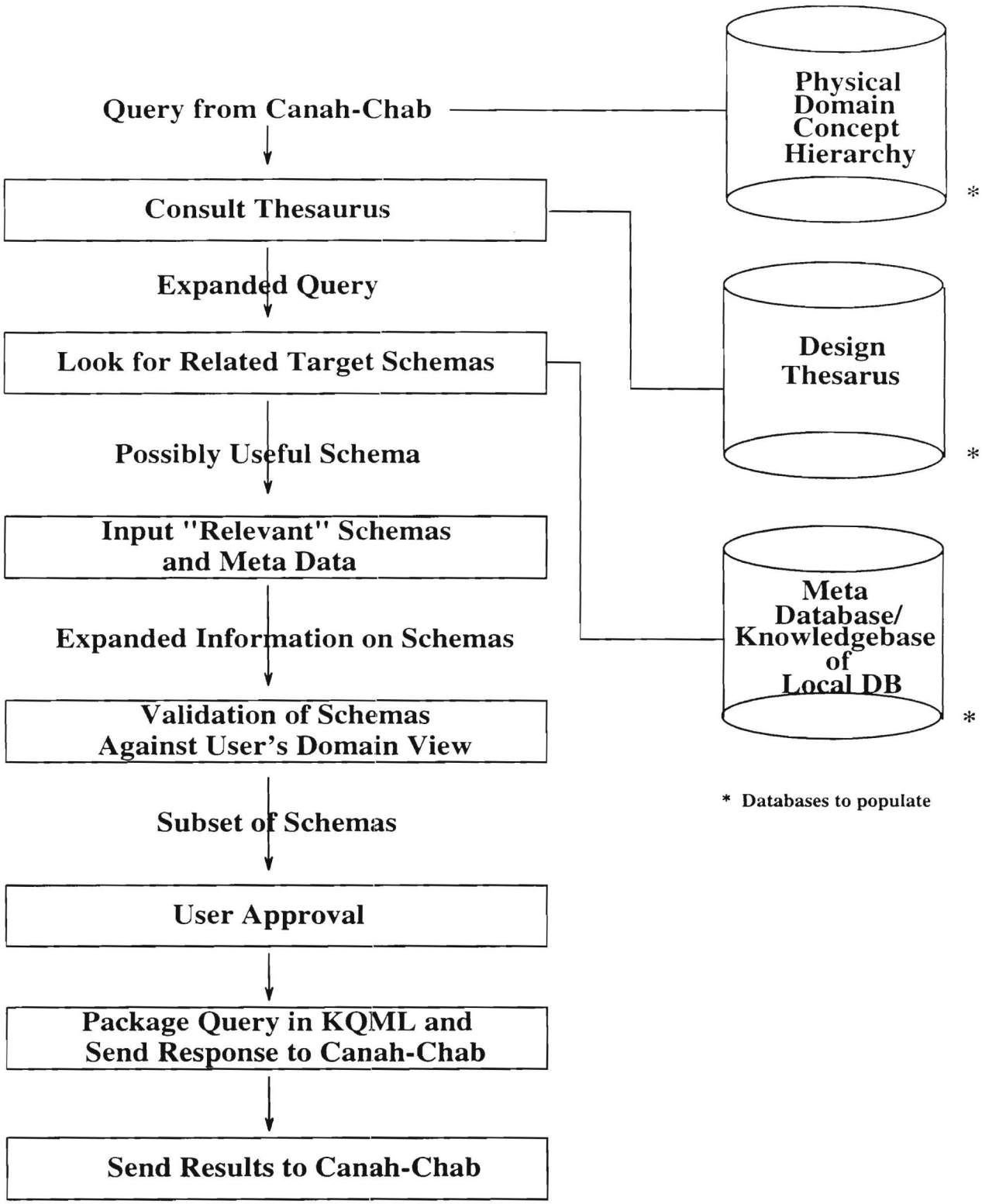


Fig. 2: Query Processing Scenario in the EDM

to information organized by site managers (similar to gopher), but users suffer from the "hypertext navigation problem" which creates difficulties in locating specific information and keeping track of where they are in the web of hypertext documents over time.

Several problems exist for the tools mentioned above. First, the tools access a particular type of data (e.g. Archie only finds source code). If a manual exists for a particular application whose source code is found by Archie, the user is not informed. Second, the tools lack relativism because the users must access the data in the manner dictated by the site manager (e.g. in WWW the data is explicitly organized by hyperlinks). Third, some of the applications require a particular site organization (e.g. Gopher requires a specific directory structure). If a site has information but no desire to organize it, a gopher search may not find the relevant information at that site. Fourth, the query processors provide little organization to the data (e.g. Archie does not organize its source code references by application type, instead all applications with a substring match on the query are returned). For these reasons, the Internet environment provides a true testbed for large scale, heterogeneous information source integration.

We propose a query processing application which, using the native internetwork capabilities, provides a single interface for accessing all types of data regardless of source or format. The following list proposes some of the necessary extensions to the EDM:

- The system should perform automated "net surfing" to create an intelligent index of each data store's information. The intelligence of the index lies in the ability to discern between types of data (audio, text, source, etc), utilize an indexing methodology tailored to the particular data type (e.g. organize keywords of a text document by the document section), and facilitate determination of an object's relevance for a query based on the knowledge of the user's interests and technical expertise. This should require no a priori knowledge of the individual data site organization. Work is being done at the Georgia Institute of Technology in intelligent text document processing and work has been done at IBM Almaden Research Center in file classification[Vee95a]. Extensive work has been done on parsers for the various document types (e.g. html, LaTeX) on the Internet.

- The problem of data overload may result from this large scale integration. Our query processor should utilize user profiles so that only data of specific relevance and technical difficulty will be derived. Unfortunately, the user profile method of data overload reduction may eliminate relevant documents. To deal with this problem, the user needs feedback from the query processor in the form of a description of what information is/is not being considered and an explanation of why. Work in explanation is part of the Canah-Chab System[Goe93].

- Keyword searches should not be limited by the vocabulary of the query; instead, a thesaurus should be used to consider synonyms. This may result in synonym overload so user profiles should also be used in pruning the list of synonyms.

- The user is assumed to be "browsing" the available information; therefore, the query interface should provide reformulation capabilities. Reformulation techniques include iterative query alteration and positive/negative feedback from the user[Vee95b].

- The system should attempt automated knowledge acquisition to provide a better understanding of indexed objects and to find other available data stores. The following list orders levels of object knowledge in ascending complexity:

  ID Knowledge - System only knows site assigned ID of object (e.g. filename)

  Content Knowledge - System knows information about object content (e.g. keywords for text)

  Description Knowledge - System knows content knowledge and an external specification of the object.

Interrelational Knowledge - System knows all of the above and interobject relationships (e.g. papers about cancer research grouped together).

- The system should be extensible with respect to "plugging-in" different types of data indexing components and user profiles. Additionally, the system should transparently handle adding/subtracting participating sites. Utilities already exist for component indexing including parsers for various document types, image recognition utilities, etc.
- Different server systems should be able to exchange information and knowledge. Work in KQML at the University of Maryland facilitates knowledge interchange even with differing ontologies[Cha92].
- Objects must be described in terms of a nested model. For example, a document may be composed of sections which are composed of text, subsections, and graphics. Stanford's Object Exchange Model (OEM) provides "self-describing," nested objects[Pap94].
- The distributed control of the system leads to problems of object identity. For example, identical application source code may reside in multiple locations; therefore, the system should attempt to provide object identity to facilitate replicated object identification. Additionally, object versioning will allow the system to keep track of more recent versions of a retrieved object. A primitive form of object identification is supported in Stanford's OEM project [Pap94].
- External knowledge sources should be used to learn about objects in the system. For example, the query processor could inspect newsgroups or look at the manner in which objects are used in WWW to acquire knowledge about the objects and their relationships. Primitive forms of natural language understanding and concept derivation techniques may be used.
- Use of existing query systems should be considered (e.g. use WAIS server to augment search).
- Special consideration should be given to optimization including reuse of retrieved data[Don93].

6 Conclusion

We have presented a framework for research in the area of intelligent, large scale integration of information sources. Clearly, much more work needs to be done before any of the detailed functionality can be implemented. We believe that much of the research into the necessary technology has begun, and the main task lies in tailoring these technologies to the needs of large scale integration and applying them in a prototype environment. We intend to further study the concepts presented above in order to develop a flexible and extensible scheme for integrating information from heterogeneous sources. Although we wish to experiment by applying our research in the area of augmenting intelligent device design in engineering, the applicability of this technology obviously extends beyond the engineering domain.

References

[Are] Yigal Arens, Chin Chee, Chun-Nan Hsu, and Craig A. Knoblock. Retrieving and integration data from multiple information sources. To appear in International Journal on Intelligent and Cooperative Information Systems.

[Are94] Yigal Arens, Chin Chee, Chun-Nan Hsu, Hoh In, and Craig A. Knoblock. Query processing in an information mediator. ISI Technical Report, 1994.

[Bat86] C. Batini, M. Lenzernini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):325–364, Dec. 1986.

[Bec89] Howard W. Beck, Sunit K. Gala, and Shamkant B. Navathe. Classification as a query processing technique in the CANDIDE semantic data model. In *1989 IEEE Conference on Data Engineering*, pages 572–581. IEEE, 1989.

[Bor94] Alexander Borgida. Description logics in data management. Technical report, Rutgers University, July 1994.

[Bra85] R. Brachman and G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[Bri94] David Brill. *Loom Reference Manual (Version 2.0)*. ISX Corp, October 1994.

[Cha92] Hans Chalupsky, Tim Finin, Rich Fritzson, Don McKay, Stu Shapiro, and Gio Weiderhold. An overview of KQML: A knowledge query and manipulation language. Technical report, KQML Advisory Group, April 1992.

[Don93] Michael J. Donahoo. *Integration of Information in Heterogeneous Library Information Systems*. Master's thesis, Baylor University, May 1993.

[Goe93] Ashok K. Goel, Andres Garza, Nathalie Grue, M. Recker, and T. Govindaraj. Beyond domain knowledge: Towards a computing environment for the learning of design strategies and skills. Technical report, College of Computing, Georgia Tech, 1993.

[Lit90] Witold Litwin, Leo Mark, and Nick Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22(3):267–293, September 1990.

[Mar85] Leo Mark. *Self-Describing Database Systems - Formalization and Realization*. PhD thesis. Computer Science Department, University of Maryland, 1985.

[Nav91] Shamkant Navathe, Sunit K. Gala, and Seong Geum. Application of the CANDIDE semantic data model for federations of information bases. In *Invited paper, COMAD '91*. Bombay, India, December 1991.

[Nav95] Shamkant B. Navathe and Ashoka N. Savasere. A practical schema integration facility using an object-oriented model. To be published in *Object Oriented Multidatabase Systems: A Solution for Advanced Applications* (O. Bukhres and A. Elmagarmid, eds), Prentice-Hall, January 1995.

[Pap94] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. Stanford University, Department of Computer Science, Technical Report, 1994.

[Par93a] Paramax System Corporation. *Computer System Operator's Manual for the Cache-Based Intelligent Data Interface of the Intelligent Database Interface*, revision 2.3 edition, Feb. 1993.

[Par93b] Paramax Systems Corporation. *Software Design Document for the Loom Interface Module (LIM) of the Cache-Based Intelligent Database Interface*, revision 2.0 edition, Jan. 1993.

[Sav91] Ashoka Savasere, Amit Sheth, Sunit Gala, Shamkant Navathe, and Howard Marcus. On applying classification to schema integration. In *First International Workshop on Interoperability in Multidatabase Systems*, pages 258–261. IEEE Computer Society, IEEE Computer Society Press, April 1991.

[She90] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183-236, September 1990.

[She93] Amit P. Sheth, Sunit K. Gala, and Shamkant B. Navathe. On automatic reasoning for schema integration. *International Journal of Intelligent and Cooperative Information Systems*, 2(1):23-50, 1993.

[Spe88] R. Speth, editor. *Global View Definition and Multidatabase Languages - Two Approaches to Database Integration*. Amsterdam: Holland, April 1988.

[Vee95a] Aravindan Veerasamy, Scott Hudson, and Shamkant Navathe. Visual interface for textual information retrieval systems. To appear in Proceedings of IFIP 2.6 Third Working Conference on Visual Database Systems, Lausanne, Switzerland. Springer Verlag. March 1995.

[Vee95b] Aravindan Veerasamy and Shamkant Navathe. Querying, navigating and visualizing an online library catalog. Submitted for Publication, January 1995.

[Wei92] Gio Weiderhold. Mediators in the architecture of future information systems. *IEEE Computer*, pages 38–49, March 1992.

[Wei93] Gio Weiderhold. Intelligent integration of information. In Arie Segev, editor, *ACM SIGMOD International Conference*, volume 22, pages 434–437. ACM, ACM Press, June 1993.

[Wha93] Whan-Kyu Whang, Sharma Chakravathy, and Shamkant B. Navathe. Heterogeneous databases: Toward merging and querying component schema. *Computing Systems*, 6(3), August 1993. (a Univ. of California Press publication).

Rule Based Database Integration in HIPED : Heterogeneous Intelligent Processing in Engineering Design

Shamkant B. Navathe Sameer Mahajan Edward Omiecinski

College of Computing,
Georgia Institute of Technology,
Atlanta, GA 30332-0280, USA.
{sham,sameer,edwardo}@cc.gatech.edu

Abstract

In this paper [1] we describe one aspect of our research in the project called HIPED, which addressed the problem of performing design of engineering devices by accessing heterogeneous databases. The front end of the HIPED system consisted of interactive KRITIK, a multimodal reasoning system that combined case based and model based reasoning to solve a design problem. This paper focuses on the backend processing where five types of queries received from the front end are evaluated by mapping them appropriately using the "facts" about the schemas of the underlying databases and "rules" that establish the correspondance among the data in these databases in terms of relationships such as equivalence, overlap and set containment. The uniqueness of our approach stems from the fact that the mapping process is very forgiving in that the query received from the front end is evaluated with respect to a large number of possibilities. These possibilities are encoded in the form of rules that consider various ways in which the tokens in the given query may match relation names, attrribute names, or values in the underlying tables. The approach has been implemented using CORAL deductive database system as the rule processing engine.

1 Introduction

Heterogeneity of databases is becoming a necessary factor to contend with in the design of new applications because of the proliferation of database management systems that used diverse data models over the last three decades. Among widely implemented data models we have the hierarchical, network, relational and object oriented data models. A large body of work exists that deals with the mapping of these models among one another (e.g. see the mapping of models using the entity relationship model as an intermediate model in [1] [3]. While vendors are also providing middleware solutions to draw data from these legacy systems, the semantic problems of resolving, naming, scale, structure etc. that were pointed out several years ago [5] [6] still remain. The purpose of the present research was to develop a technique to dealing with the semantic differences in data by taking a flexible rule based approach. Another goal of the project was to tie a set of heterogeneous databases to an "intelligent front end application" which would make requests for data without any knowledge of the schemas of the target databases. To limit the degree of difficulty we assume that we are dealing with data in relational databases only. This assumption is reasonable in the sense that of the data is coming from a hierarchical or a network DBMS, we can first convert the schema to a relational one before treating it for purposes of integration.

The database integration problem we discuss here is couched in the context of engineering design which, like any other design application, relies on extracting data from existing databases containing material data, components, existing designs etc. The exact context and the application scenario will be explained in the next section.

We assume that relevant data for the design application is stored in relations (tables) whose schemas are available at "design time" to construct a rule-base. It is conceivable that to support large scale engineering designs, data from a variety of databases, i.e., from multiple schemas would be required. To facilitate integration of data among these databases we assume that the "correspondances", i.e., the similarities and differences among the (meaning of) attributes is encoded in the form of rules. Furthermore, for our application context, the front end of HIPED issues certain queries looking for relevant design information. We show in this paper how a query may have several interpretations, each one of which is encoded in the form of rules again.

Because of these two kinds of rules involved in the integration approach we have termed our approach a rule based approach to database integration. The present approach is an improvement over previous approaches where we handled integration by using the correspondance information to derive the process [2] [6] [7] [8].

2 Application Context

In this section we will provide the overall architecture of the HIPED system and point out the need for heterogeneous database processing which will be de-

[1] To appear in the Proceedings of International Symposium on Cooperative Database Systems for Advanced Applications, Heian Shrine, Kyoto, Japan, World Scientific Press, 1996.

scribed and illustrated in the next two sections.

2.1 Overall Architecture of HIPED

Our main objective in the HIPED project is to build and demonstrate an intelligent interface to a set of (possibly autonomous) information sources including structured databases, knowledge bases, and unstructured data. The approach we have selected involves the development of a mediator which utilizes meta-knowledge of the underlying information stores to aid a user in browsing data or to enable an application front-end to retrieve specific relevant information for problem solving.

The overall architecture of HIPED is described in Figure 1. We look at only the "Database Bachend" in this paper. The data is organized at two levels namely, (1) the metadata repository : consisting of information about various databases and tables in them and (2) the actual data which is distributed in various heterogeneous databases. This organization reduces the data to be dealt with at the first level to get to the appropriate database(s) and table(s). It also allows heterogeneity in the various databases involved. The Querying Interface is as described in section 3.1. The "data" together with its "wrapper" forms a database system. "Wrapper" simply defines the access methods to the data for reading purposes. A wrapper can be designed for each target database management system. A user query would be translated into the corresponding query, as understood by the corresponding "wrapper", for each of the relevant tables. This query would then be routed to the corresponding database, that contains this table. The metadata repository is consulted in determining these relevant tables and finding the corresponding database. The user would get the result, obtained after running the query against the table through the concerned "Output Data" channel(s).

2.2 Interactive KRITIK Front End

We developed the HIPED architecture by assuming a frontend system called Interactive Kritik [4]. This system is a multimode reasoning system which works like a design assistant for the design of devices such as acid coolers, electrical devices. In its current form the system uses "hard-wired" knowledge in the form of LISP data structures. The goal was to extend the capability of interactive Kritik to make it scalable to real-life design problems by incorporating databases of relevant design data as the back end. We therefore abstracted different forms of generic query types which would be used as requests to the back end. By coupling an intelligent front end application to a set of heterogeneous databases, we can thus extend the scope of problem solving by a large measure. For engineering device design, the above front end generates a number of requests for data from the underlying design databases such as design prototypes, properties of devices and components, material data, design specifications and tolerances etc. For illustrative purposes we have chosen five generic types of queries that are most commonly presented by the front end. They will be explained in detail in the following section.

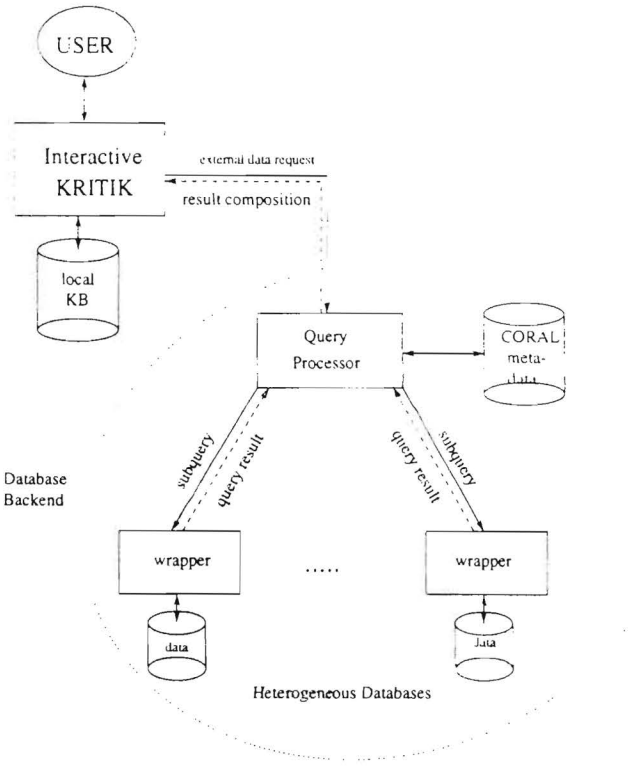


Figure 1: The High Level View of HIPED

3 Rule Based Approach to Database Integration

As explained earlier the main contribution of this research is the use of the two types of rules to accomplish access to the underlying heterogeneous information sources. The first set of rules deals with establishing various types of relationships among relation names and among attribute names across databases. The second set deals with the interpretation of queries from the front end so that various possible mappings to the interface of underlying target databases may be considered. We will explain both these types of rules when we discuss the generic queries and their mappings.

3.1 Five generic types of queries

The user is assumed to use this system as an Engineering Database for device design. Let us limit the application domain for illustrative purposes. We assume that during the design process, he would typically like to find components that satisfy his requirements (e.g. batteries with voltage rating higher than 10V and cheaper than $10). Keeping this user's perspective in mind, the Engineering data is thought to be made up of various "Prototypes". Each Prototype has various "Properties". Each Property takes up some "Value" for every Prototype. We can compare the Values of various properties using the relations : ==, <, >, <=, >=, <> etc. The queries can be classified into the following five generic types,

1. (Prototype <proto_name>) : here the user is looking for all the prototypes identified by "proto_name". It is implicit that the user wants to see the various values for various properties (attributes) of these prototypes.

2. (Property <prop_name>) : the user is interested in all the prototypes having the specific Property identified by "prop_name". It is implicit that the user wants to see the values taken by this property for the various prototypes, that would be listed.

3. (Prototype <proto_name>)
(Property <prop_name>) : the user wants to see all the prototypes identified by "proto_name" and having property identified by "prop_name". It is implicit that the user also wants to see the corresponding value that the property takes for the particular prototype.

4. (Prototype <proto_name>)
(Property <prop_name>)
(Value<value>) (Rel-op <op>) : the user is interested in prototypes identified by "proto_name" having a property identified by "prop_name". In addition to this he wants only those prototypes for which the property takes a value which is related to the given "value" or a constant in the query by the operator "op" (i.e. it is equal to "value" or greater than "value" etc.)

5. (Property <prop_name>) (Value <value>)
(Rel-op <op>) : the user is interested in all the prototypes for which the property identified by "prop_name" takes a value which is related to the given "value" by the operator "op.

Data is distributed among various databases and various tables in each of those databases. The only assumption that we make about any database system is that it has an SQL access method. It is a reasonable assumption and is made to contain the complexity of the problem.

The system needs to find out which databases and which tables in these databases have the relevant data to answer a particular query. It then translates the query into a corresponding SQL query for every table. This SQL query is run against that table to get an answer. As we made an assumption of a uniform SQL interface to all the databases, we can simply translate a request for data into a set of SQL queries in each of these cases.

3.2 Rules for Interpretation of Queries

For better understanding of the following discussion, let us take up an example query. Let the four components of the query be,
(Prototype Battery) (Property Voltage)
(Value 10) (Relation ==).
As there can be various tables with different schema, we need to run this query with only those tables that might give meaningful results for the query. We can easily observe that any of "Prototype", "Battery", "Property" and "Voltage" can be a table or a column of a table. The "Battery" and "Voltage" can also be values in the columns (e.g. those labeled as "Prototype" and "Property" respectively). Of course there are a lot of dependencies amongst these components - e.g. if "Prototype" is a table then "Battery" has to be a column of this table. On the other hand if there is a table called the "Battery", then we are looking for values in the column "voltage" or "volts" - so that the query would generate meaningful results with the table. Now we take up an example query for each of the five types listed above. For every query we list the possible interpretations according to our scheme.

1. (Prototype Battery). The user typically means that he wants all the batteries with their properties and their corresponding values. Hence we will have to run this query against all the tables which,

   - are equivalent to "Prototype Table" and have a column equivalent to "Battery" or
   - are equivalent to "Battery Table"
   - have a column equivalent to "Prototype" (and only the tuples with Prototype as "Battery" would be considered).

   if and only if these tables have columns equivalent to "Property" and "Value" each.

2. (Property Voltage). The user is interested in listing all the Prototypes having "Voltage" as their one of the Properties. The Values of these Properties would also be significant from his standpoint. Hence we consider all the tables which,

   - are equivalent to "Property Table" and have a column equivalent to "Voltage" or
   - are equivalent to "Voltage Table"
   - have a column equivalent to "Property" (and only the tuples with Property as "Voltage" would be considered).

   if and only if they have "Prototype" equivalent column.

3. (Prototype Battery) (Property Voltage). The user wants all the batteries with special interest in their voltages. Hence we will run the query against all the tables which,

   - are equivalent to "Prototype Table" and have "Battery", "Property" and "Value" equivalent columns and we would be interested only in the tuples having an entry of "Voltage" in the "Property" equivalent column or
   - are equivalent to "Prototype Table" and have "Battery", "Voltage" equivalent columns or
   - are equivalent to "Battery Table" and have "Property" and "Value" equivalent columns. We would be interested only in those tuples having Property "Voltage" or

- are equivalent to "Battery Table" and have a column equivalent to "Voltage" or
- are equivalent to "Property Table" and have columns equivalent to "Voltage", "Prototype" and "Value". We would be interested in tuples with Prototype as "Battery".
- are equivalent to "Property Table" and have "Voltage" and "Battery" equivalent columns.
- are equivalent to "Voltage Tables" and have "Prototype" and "Value" equivalent columns. We would look for only tuples with Prototype as "Battery".
- are equivalent to "Voltage Table" with "Battery" and "Value" equivalent columns.
- have "Prototype" and "Property" equivalent columns as far as they have "Value" equivalent column. Only the tuples with Prototype as "Battery" and Property as "Voltage" would be considered.

4. (Prototype Battery) (Property Voltage) (Value 10) (Relation ==). Here the interest is indicated in all the batteries having Voltage as "10". The query can be run with all the tables as indicated as above with an added constraint that only those tuples which have an entry of "10" in the "Voltage" or "Value" column - whichever is applicable - (Note the table can have only one of these columns at a time) will be considered.

5. (Property Voltage) (Value 10) (Relation ==). All the Prototypes having voltage of "10" are being considered. Thus all the tables that,

   - are equivalent to "Property Table" and have a column equivalent to "Voltage"
   - are equivalent to "Voltage Table" and have a column equivalent to "Value"
   - have "Property" and "Value" equivalent columns along with "Prototype" column. (only tuples with Property "Voltage" and Value "10" would be taken into consideration).

   would be considered if and only if they have a column equivalent to "Prototype". All the tuples with "Voltage" or "Value" being 10 would be taken into account.

3.3 Rules to establish Data Correspondance

We need to relate various attributes and tables, within and across databases. The relationship could be of equivalence, subsumption, overlap, disjointness or containment. The relationship between attributes needs to be supplied by the schema developer. e.g. Attributes called "volt" and "voltage" in different tables are actually equivalent. The relationship between tables can either be supplied or can be deduced by the relationships of their individual attributes. A simple deduction rule can be that two tables are equivalent if all their attributes are equivalent.

4 Use of CORAL for rule representation and query processing

The metadata is stored in the form of CORAL [10] [11] facts and rules. CORAL is a deductive database system which stores data as facts and rules, and allows for that data to be queried. By using CORAL the mediator can decide which database(s) and table(s) are useful in answering any given query. In particular, CORAL is used in deriving relationships like equivalence; between attributes, tables and databases. Any creation, deletion or modification of a table results in a change in the metadata repository. This dynamic behavior can be easily captured by CORAL. In essence, CORAL provides us with the facility for database integration through the facts and rules specified about tables and databases. However, this integration can be considered implicit rather than explicit since no global conceptual schema is explicitly formed. Also the C++ interface provided by CORAL makes writing general purpose programs easy.

We explain the implementation with the help of an example. One more sample system for a single database environment is given in Table 5. Some sample input queries and the corresponding output SQL queries are shown in Tables 6 and 7 respectively.

4.1 A Simple Example

Consider the query,

```
(Prototype Battery) (Property Voltage).
```

Let us assume that there are two databases - db1 and db2. Let db1 have tables : Table 1 and Table 2. and

CompNo	Prototype	Property	Value
B101	Battery	Voltage	10
M101	Motor	Voltage	10
B110	Battery	Voltage	100
B111	Battery	Current	100
	⋮		

Table 1: "Components" Table in db1

le db2 have Table 3 and Table 4.

We observe that according to the discussion in section 3.2 only the tables in db1 would produce meaningful results with the query under consideration.

BatteryNo	Voltage
B101	15
B102	30
⋮	

Table 2: "Battery" Table in db1

BatteryNo	Current
B101	15
B102	30
	⋮

Table 3: "Battery" Table in db2

BatteryNo	Supplier No
B101	4567
B102	4568
	⋮

Table 4: "Supplier" Table in db2

4.2 Schema Representation

It is stored as CORAL facts and rules. The advantage of such a storage is that we can utilize the strong deductive power of CORAL (e.g. deducing equivalence of attributes, equivalence of tables etc.). The various components of the repository are described below.

- First we list all the tables in all the databases as CORAL facts :

```
% For the first database, db1.
belongsTo(components,db1).
belongsTo(battery,db1).

% For the second database, db2.
belongsTo(battery,db2).
belongsTo(supplier,db2).
```

- Then we list attributes of individual tables as CORAL facts. The first argument of these predicates is the database name. It is so because the same table may have different attributes in different databases. e.g. the "battery" table in the two databases "db1" and "db2" as shown below.

```
% for db1
hasAttribs(db1,components,
   [compName,prototype,property,value]).
hasAttribs(db1,battery,[bName,voltage]).

% for db2
hasAttribs(db2,battery,[bName,current]).
hasAttribs(db2,supplier,[bName,sName]).
```

- We also need facts to list what attributes are equivalent. The equivalence of tables can be either given by facts or can be deduced by the rules (e.g. two tables with equivalent attributes are equivalent). But we do not need them in this particular example.

- To find whether a table has a particular attribute in a given database we define a CORAL rule as.

```
module isAttrib.
export isAttrib(bff).
isAttrib (Db,Table,Attri) :-
        hasAttribs(Db,Table,Attribs),
        iselem (Attri,Attribs).
end_module.

% Module ''iselem'' is defined for the
% sake of completeness.
module iselem.
export iselem(bb).
@pipelining+.     % Solve in a
                  % top-down fashion

iselem(X, [X|_]).
iselem(X, [_|Z]) :- iselem(X,Z).
end_module.
```

4.3 Sample Query Mapping Algorithm

The mapping of input requests into SQL queries is done according to the scheme suggested in section 3.2. We use the C++ interface of CORAL for this matter. In fact, an imperative interface (e.g. in C) would have been enough for the purpose. We check for the various conditions given in the scheme and generate the appropriate SQL queries for the existing tables. We run through the algorithm for the example query under consideration,

```
begin
For every ''table'' equivalent to
''prototype table''
  for every attribute equivalent
  to ''battery'', say attrib1
    for every attribute equivalent
    to ''property'', say attrib2
      if ''table'' has ''attrib1''
      as well as ''attrib2''
        for every attribute
        equivalent to ''value'',
        say attrib3
          if ''table'' has attrib3
            select the corresponding
            database and fire SQL query,
            SELECT * FROM table
            WHERE attrib2 == voltage or
                  some equivalent value.
            goto next table
        for every attribute equivalent
        to ''voltage'', say attrib4
          if ''table'' has attrib4
            select the corresponding
            database and fire SQL query,
            SELECT * FROM table
```

```
For every ''table'' equivalent to
''battery table''
  for every attribute equivalent
  to ''voltage'', say attrib1
    if ''table'' has attrib1
        select the corresponding
        database and fire SQL query,
        SELECT * FROM table
        goto next table
  for every attribute equivalent
  to ''property'', say attrib2
    for every attribute equivalent
    to ''value'', say attrib3
      if ''table'' has attrib2
      and attrib3
        select the corresponding
        database and fire SQL query,
        SELECT * FROM table
        WHERE attrib2 == voltage or
              some equivalent value.

For every ''table'' equivalent
to ''property table''
  for every attribute equivalent
  to ''voltage'', say attrib1
    for every attribute equivalent
    to ''prototype'', say attrib2
      if ''table'' has ''attrib1''
      as well as ''attrib2''
        for every attribute
        equivalent to ''value'',
        say attrib3
          if ''table'' has attrib3
            select the corresponding
            database and fire SQL query,
            SELECT * FROM table
            WHERE attrib2 == battery or
                  some equivalent value.
            goto next table
        for every attribute equivalent
        to ''battery'', say attrib4
          if ''table'' has attrib4
            select the corresponding
            database and fire SQL query,
            SELECT * FROM table

For every table equivalent to
''voltage table''
  for every attribute equivalent
  to ''battery'', say attrib1
    if ''table'' has attrib1
      select the corresponding
      database and fire SQL query,
      SELECT * FROM table
      goto next table
  for every attribute equivalent
  to ''prototype'', say attrib2
    for every attribute equivalent
    to ''value'', say attrib3
      if ''table'' has attrib2
      and attrib3
        select the corresponding
        database and fire SQL query,
        SELECT * FROM table
        WHERE attrib2 == battery or
              some equivalent value.

For every table having columns
equivalent to each of
prototype, property and value
  select the corresponding
  database and fire SQL query
  SELECT * FROM table
  WHERE prototype equivalent column
          == battery equivalent value
        AND
        property equivalent column
          == voltage equivalent value
end
```

4.4 The Result

Let us say that the wrapper of db1 can handle SQL queries. In that case we first select that database and then simply run a query,

```
SELECT *
FROM components
WHERE prototype == ''battery''
        AND property == ''voltage''
```

against the first ("components") table in the database. We take similar actions for the other table in (possibly various) databases. The other query in this case would be,

```
SELECT *
FROM battery
```

again with the same database namely, db1. The result is presented to the user as displayed by the corresponding "wrapper".

5 Conclusions and Future Work

In this paper we illustrated the implementation of a rule-based database integration scheme by considering two types of rules : (1) Rules to establish the "correspondence" among underlying component databases and (2) Rules to interpret data requests in an "open-ended" fashion where no knowledge of the component database schemas is expected from the application front end. We also described an interface to heterogeneous databases in which a user may directly access the back end data by making use of the rules of data correspondance and an SQL-like syntax for the queries.

The system makes an assumption that all the databases involved provide an SQL interface. This condition can be relaxed. In this case we need to generate different queries, as understood by each of the databases involved. This work was predicated on the assumption that the data relevant to our application was stored in relational tables. An extension of the present work involves relaxing this assumption and illustrating the utility of the approach by actually providing wrappers for hierarchical and network databases and sequential files. That would establish

```
% CORAL facts
isTable(battery).
hasAttribs(battery,
   [bname,voltage,current,life]).

isTable(compTable).
hasAttribs(compTable,
   [no,prototype,property,value]).

isTable(dummy).
hasAttrib(dummy,[prototype,property]).

isTable(prototype).
hasAttribs(prototype,
   [motor,property,value]).

isTable(motor).
hasAttribs(motor,[property,value]).

isTable(property).
hasAttribs(property,
            [rps,prototype,value]).

isTable(rps).
hasAttribs(rps,[prototype,value]).

isTable(voltage).
hasAttribs(voltage,[battery,value]).

% CORAL rules

module isAttrib.
export isAttrib(ff).
isAttrib (X,Y) :- hasAttribs(X,Z),
                  iselem (Y,Z).
end_module.
```

Table 5: A Single Database System

```
prototype battery property voltage
prototype battery property current
prototype motor property rps
prototype sheet property size
```

Table 6: Sample Input Queries

```
++++++++ for the first data request ++++++
SELECT * FROM battery;
SELECT * FROM voltage;
SELECT * FROM compTable
WHERE prototype == battery
        AND property == voltage;
++++++++ for the second data request +++++
SELECT * FROM battery;
SELECT * FROM compTable
WHERE prototype == battery
        AND property == current;
++++++++ for the third data request ++++++
SELECT * FROM prototype
WHERE property == rps;
SELECT * FROM motor
WHERE property == rps;
SELECT * FROM property
WHERE prototype == motor;
SELECT * FROM rps
WHERE prototype == motor;
SELECT * FROM compTable
WHERE prototype == motor
        AND property == rps;
+++++++ for the fourth data request +++++
SELECT * FROM compTable
WHERE prototype == sheet
        AND property == size;
```

Table 7: The corresponding SQL queries

the practical utility of the approach in a significant way. The next step would be to work on a query optimization by introducing a stage after the query interpretation phase to evaluate possible orderings of sub queries and cross subquery reduction of redundant processing.

From the engineering design standpoint, the problem horizon can be extended to include additional types of design problems. The current implementation can be initially enhanced by considering additional types of design queries.

Currently only the individual tables are checked to see whether they provide satisfactory data to answer a particular query. But it is possible that two or more tables taken separately do not have enough information to answer a query. At the same time, when taken together (e.g. their join), they provide data to answer the query. Consider that there are two tables - which might be in the same database or in different databases - one with columns "Component Number" and "Prototype". The other with columns "Component Number" and "Voltage". Then neither of them provides enough information for the query,

(Prototype Battery) (Property Voltage)

But their equijoin with the additional condition of "Prototype == Battery" for the tuples is of interest to us. The extended solution can exhaustively take care of all such cases.

In essence, the overall rule based approach appears promising in the context of Navathe's long standing

investigation of the database integration problem [5] [6] [7] [8] [9].

Acknowledgement

We would like to thank Ashok Goel and William Murdock for their work on Interactive KRITIK. The work crystallized the HIPED front end to our system. We are also grateful to Jeff Donahoo, Ashok Savasere and Byong-soo Jeong for initializing the implementation work. The idea of using CORAL as a deductive database system was generated by Ed Omiecinski. The implementation was completed by Siddharth Bajaj. Support from ARPA grant no. F33615-93-1-1338 is greatly appreciated. Prof. Navathe's work is also partially supported by Center of Excellence in Information Science, Clark Atlanta University, Contract No. OSP-93-09-400-002.

References

[1] C. Batini, S. Ceri, and S.B. Navathe, **Conceptual Database Design: An Entity Relationship Approach**, Benjamin Cummings, August 1991, 470 pp.

[2] C. Batini, M. Lenzerini and S.B. Navathe, A Comparative Analysis of Methodologies for Database Schema Integration, *ACM Computing Surveys*, 18, 4, December 1986, pp. 323-364.

[3] R. Elmasri, S Navathe, **Fundamentals of Database Systems**, Addision Wesley Computer Science, 2nd Edition, 1994.

[4] Ashok Goel, Andres Gomez, Nathalie Grue, William Murdock, Margaret Recker, and T. Govindaraj. Design Explanations in Interactive Design Environments. In *Proc. Fourth International Conference on AI in Design*, Palo Alto, June 1996.

[5] J. Larson, S. B. Navathe, and R. Elmasri A Theory of Attribute Equivalence in Databases with Application to Schema Integration, *IEEE Transactions on Software Engineering*, Vol. 15, No. 4, April 1989.

[6] S.B. Navathe, R. Elmasri and J.A. Larson, Integrating User Views in Database Design, *IEEE Computer*, Vol. 19, No. 1, January 1986, pp. 50-62.

[7] S. B. Navathe and A. Savasere, "A Practical Schema Integration Facility using an Object Oriented Approach," *Multidatabase Systems (A. Elmagarmid and O.Bukhres, Eds.)*, Prentice Hall, 1996.

[8] S. Prabhakar, J. Srivastava, S.B. Navathe, et al., Federated Autonomous Databases: Project Overview, *Proceedings of the International Conference on Interoperability in Multidatabase Systems (IMS'93)*, Vienna, Austria, April 19-20, 1993.

[9] A. Sheth, S.K. Gala, S.B. Navathe, *On Automatic Reasoning for Schema Integration, Int. Journal of Intelligent Co-operative Information Systems*, Vol. 2, No.1, March 1993.

[10] R. Ramakrishnan, D. Srivastava and S. Sudarshan, CORAL: Control, Relations and Logic, *Proceedings of the International Conference on Very Large Databases*, 1992.

[11] R. Ramakrishnan, D. Srivastava, S. Sudarshan and P. Seshadri, Implementation of the CORAL deductive database system, *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1993.

From Data to Knowledge: Method-Specific Transformations*

Michael J. Donahoo,
J. William Murdock,
Ashok K. Goel,
Shamkant Navathe,
and Edward Omiecinski
College of Computing
Georgia Institute of Technology

Keywords: Intelligent Information Systems, Integration of Knowledge and Database Systems

Abstract

Generality and scale are important but difficult issues in knowledge engineering. At the root of the difficulty lie two hard questions: how to accumulate huge volumes of knowledge, and how to support heterogeneous knowledge and processing? One answer to the first question is to reuse legacy knowledge systems, integrate knowledge systems with legacy databases, and enable sharing of the databases by multiple knowledge systems. We present an architecture called HIPED for realizing this answer. HIPED converts the second question above into a new form: how to convert data accessed from a legacy database into a form appropriate to the processing method used in a legacy knowledge system? One answer to this reformed question is to use method-specific transformation of data into knowledge. We describe an experiment in which a legacy knowledge system called Interactive Kritik is integrated with an ORACLE database using IDI as the communication tool. The experiment indicates the computational feasibility of method-specific data-to-knowledge transformations.

*Contact: Ashok K. Goel, College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, Georgia 30332-0280, USA; email: goel@cc.gatech.edu; phone: (404) 377-3732

1 Motivations, Background, and Goals

Generality and scale have been important issues in knowledge systems research ever since the development of the first expert systems in the mid sixties. Yet, some thirty years later, the two issues remain largely unresolved. Consider, for example, current knowledge systems for engineering design. The scale of these systems is quite small both in the amount and variety of knowledge they contain, and the size and complexity of problems they solve. In addition, these systems are both domain-specific in that their knowledge is relevant only to a limited class of domains, and task-specific in that their processing is appropriate only to a limited class of tasks.

At the root of the difficulty lie two critical questions. Both generality and scale demand huge volumes of knowledge. Consider, for example, knowledge systems for a specific phase and a particular kind of engineering design, namely, conceptual phase of functional design of mechanical devices. A robust knowledge system for even this very limited task may require knowledge of millions of design parts. Thus the first hard question is this: How might we accumulate huge volumes of knowledge? Generality also implies heterogeneity in both knowledge and processing. Consider again knowledge systems for the conceptual phase of functional design of mechanical devices. A robust knowledge system may use a number of processing methods such as problem/object decomposition, prototype/plan instantiation, case-based reuse, model-based diagnosis and model-based simulation. Each of these methods uses different kinds of knowledge. Thus the second hard question is this: How might we support heterogeneous knowledge and processing?

Recent work on these questions may be categorized into two families of research strategies: (i) *ontological engineering*, and (ii) *reuse*, *integration* and *sharing* of information sources. The well known CYC project [Lenat and Guha, 1990] that seeks to provide a global ontology for constructing knowledge systems exemplifies the strategy of ontological engineering. This bottom-up strategy focuses on the first question of accumulation of knowledge: global (and domain-specific) ontologies may one day enable interactive knowledge acquisition from external sources and autonomous acquisition of knowledge through learning from experience. The second research strategy has three elements: reuse of information sources such as knowledge systems and databases, integration of information sources, and sharing of information in one source by other systems. This top-down strategy emphasizes the second question of heterogeneity of knowledge and processing, and appears especially attractive with the advent of the world-wide-web which provides access to huge numbers of heterogeneous information sources such as knowledge systems, electronic databases and digital libraries. Our work falls under this category which we will call the share-integrate-reuse (or SIR) strategy.

Various members of the SIR family of research strategies focus on different aspects of reuse, integrate and sharing. For example, KQML [Finin and Wiederhold, 1991] provides a protocol for communication among database systems, and KIF [Genesreth and Fikes, 1991] provides a meta-language for enabling translation between knowledge systems. In contrast, Brodie [Brodie, 1988] has emphasized the need for integrating knowledge systems and databases. Our work focuses on this interface between knowledge systems and databases.

McKay, Finin and O'Hare [McKay et al., 1990] have pointed out that the key question pertaining to this interface is how to convert data in a database into knowledge useful to a knowledge system. The answer to this question depends in part of the processing

method used by the knowledge system. The first generation of knowledge systems (called expert systems) demonstrated the power of production systems, which combined domain knowledge (in the form of production rules) and control of processing (forward/backward chaining). The second generation of knowledge systems used *task-specific* architectures in which both domain knowledge and control of processing are dependent on the reasoning task (e.g., [Clancey, 1985, Chandrasekaran, 1988]). The current third generation of knowledge systems emphasize *method-specific* architectures in which the domain knowledge and control of processing dependent on the processing method. For example, Brown and Chandrasekaran [Brown and Chandrasekaran, 1989] describe the generic method of plan instantiation and expansion, and Marcus, Stout and McDermott [Marcus et al., 1988] describe the general method of propose and revise. The third generation of knowledge systems are heterogeneous both in their domain knowledge and control of processing. They not only use multiple methods, each of which uses a specific kind of knowledge and control of processing, but they also enable dynamic method selection. Our work focuses on the interface between legacy databases and legacy knowledge systems of the third generation.

The issue then becomes: given a legacy database, and given a legacy knowledge system in which a specific processing method poses a particular knowledge goal (or query), how might the data in the database be converted into a form appropriate to the processing method? The form of question also indicates a possible answer: *method-specific transformation* (or MST), which would transform the data into a form appropriate to the processing strategy. The goal of this paper is outline a conceptual framework for the SIR approach and the MST technique. Portions of this framework are instantiated in an operational computer program called HIPED (for Heterogeneous Intelligent Processing for Engineering Design). HIPED integrates a third-generation knowledge system for engineering design called Interactive Kritik [Goel et al., 1996a, Goel et al., 1996b] with an external database represented in Oracle [Koch and Loney, 1995]. The knowledge system and the database communicate through IDI [Paramax, 1993]. The rest of paper is organized as follows: in the next section, we describe our conceptual framework; in the following section, we describe a specific experiment with the framework; and in the final section, we describe related work in the HIPED project and conclude the paper.

2 HIPED Architecture

To avoid the enormous cost of constructing knowledge systems, we propose the reuse of legacy knowledge and database systems, so that we can quickly and inexpensively construct large scale systems with capabilities and knowledge drawn from existing systems. To facilitate easy integration which, in effect, increases overall scalability, we restrict ourselves to making few, if any, changes to the participating legacy systems. The long-term goal is to allow a system to easily access the capabilities of a pool of legacy systems. The architecture of Figure 1 illustrates the general scheme. We describe this architecture in the following subsection by decomposing it into database, knowledge system, and user components.

2.1 Database Integration

An enormous amount of data is housed in various database systems; unfortunately, the meaning of this data is not encoded within the databases themselves. At best, the database

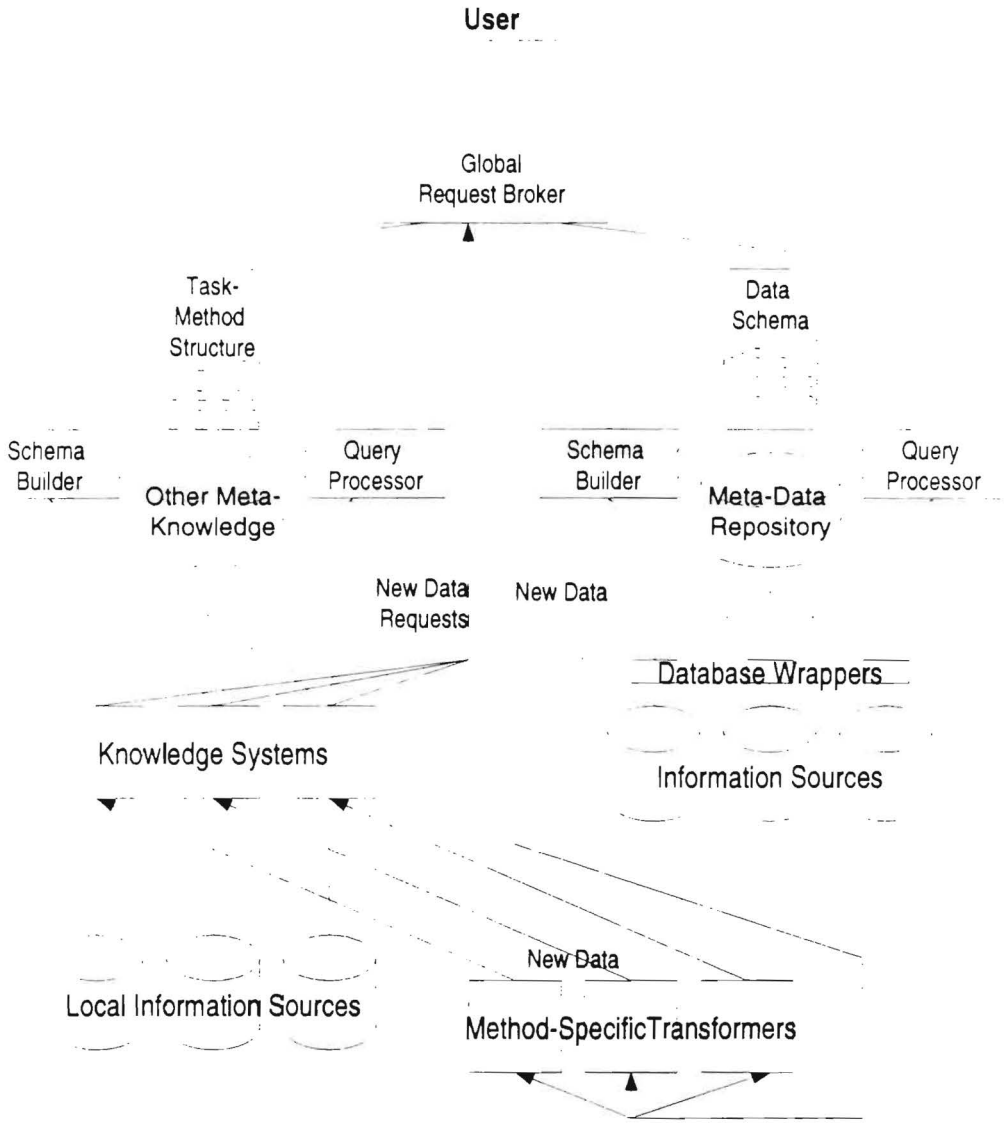


Figure 1: This figure presents a general architecture for HIPED. Arrowed lines indicate unidirectional flow of information. All other lines indicate bidirectional flow; there is a wide variety of potential execution paths through this architecture. Annotations on lines, where they occur, describe the nature of the information which flows through that line. Rectangular boxes indicate functional units and cylinders represent collections of data.

schema has meaningful names for components, but often it is difficult to infer all, if any, of the meaning of the data from the schema. This lack of metadata about the schema and a myriad of interfaces to various database systems creates significant difficulties in accessing data from various legacy database systems. Both of these problems can be alleviated by creating a single, global representation of all of the legacy data, which can be accessed through a single interface. Common practice for integration of legacy systems involves manual integration of each legacy schema into a global schema. That is, database designers of the various legacy systems create a global schema capable of representing the collection of data in the legacy databases and provide a mapping between the legacy system schemas and this global schema[Batini et al., 1986]. Clearly, this approach does not work for integration of a large number of database systems.

We propose (See the right side of figure 1) to allow the database designers to develop a metadata description, called an augmented export schema, of their database system. A collection of augmented export schemas can then be automatically processed by a schema builder to create a partially integrated global schema. We call this "partially integrated" because it is unlikely that the schema builder will be able to create a schema which correctly identifies all the relationships. The augmented export schema can be as simple as the actual database schema, allowing any database to easily participate, or as complicated as the schema builder can understand (See [Navathe and Donahoo, 1995] for details on possible components of an augmented export schema). A user can then submit queries on the partially integrated global schema to a query processor which fragments the query into queries on the local databases. Queries on the local databases can be expressed in a single query language which is coerced to the local databases query language by a database wrapper.

2.2 Knowledge System Integration

As with databases, a considerable number knowledge systems exist, each with their own abilities to perform certain tasks with various methods. Users wishing to access the capabilities of a collection of knowledge systems encounter problems of different interfaces and knowledge representations. Most knowledge systems do not provide an externally accessible description of the tasks and methods they address. As with the database system, one way to integrate legacy knowledge systems is to gather together the designers and construct an ad hoc interface with combines the capabilities of the underlying systems. Once again, this approach does not work for integration of a large number of knowledge systems.

We propose (See the left side of figure 1) to allow knowledge system designers to develop a description, called a "task-method schema," of the tasks each local knowledge system can perform[Stroulia and Goel, 1995]. In this approach, a set of knowledge systems, defined at the level of tasks and methods, are organized into a coherent whole by a query processor or central control agent. The query processor uses a hierarchically organized schema of tasks and methods as well as a collection of miscellaneous knowledge about processing and control (i.e. other meta-knowledge). Both the task-method structure and the other meta-knowledge may be constructed by the system designer at design time or built up by an automated schema builder.

2.3 Global User Access

The dichotomy of knowledge systems and database systems is irrelevant to global users. Users simply want answers and are not concern with whether the answer was provided directly from a database or derived by a process in a knowledge system. We propose the provision of a global request broker which takes a query from a user, submits the query to both knowledge and database systems and returns an integrated result. It is completely transparent to a global user how or from where an answer was derived.

2.4 Knowledge Systems as Users

The knowledge systems each have their own local repositories of data but may also find that they need information from a database or another knowledge system. When they need external knowledge, they simply contact the global request router which can either recursively call the general collection of knowledge systems to generate a response or contact the system of databases. When either a database or a knowledge system generates a response to a request from a knowledge system, the resulting answer is then sent through a method specific transformer which do whatever specific translation is needed for the particular system.

2.5 Method-Specific Transformation (MSTs)

In this paper, we are concerned with the transformation of knowledge from external sources into a form suitable for use by a knowledge system method. Recall that we do not want to alter the knowledge system, so the form of the knowledge may be very specific to the particular method which executed the query; consequently, we call this a "method-specific transformation." A naive approach to providing the necessary translations involves writing a transformation function for every permutation of knowledge system and database. That is, for each knowledge or database system issuing a query and for each system returning data, we must provide a function which translates answers from one representation to the query systems representation. Clearly, this limits the overall scalability of the system.

We propose the leverage the partially integrated global representation of the knowledge and database systems. We accomplish this by creating a method-specific transformation for each knowledge system which transforms knowledge from the partially integrated global schema into a knowledge system specific representation. Note that now the number of necessary method-specific transformations is linear with respect to the number of knowledge systems, increasing the scalability of our approach.

2.6 Information Flow

Consider a knowledge system which spawns a task for finding a battery which has a certain voltage. In addition to continuing its own internal processing, the knowledge system also submits a query to the Global Request Broker. The broker sends the query to the query processors for both integrated knowledge and database systems. The database query processor fragments the query into subqueries for the individual databases. The data derived is merged, converted to the global representation, and returned to the Global Request Broker. In the meanwhile, the knowledge query processor, using its task-method schema,

selects knowledge systems with appropriate capabilities and submits tasks to each. Solutions are converted to a common representation and sent to the Global Request Broker. The Global Request Broker then passes the output from both the knowledge and database system query processors through a method-specific transformer which coerces the data into a form which is usable by the requesting knowledge system. The resulting battery may be an actual battery which satisfies the voltage specification from a knowledge or database system information source or it may be a battery constructed from a set of lower voltage batteries by a knowledge system.

3 An Experiment with HIPED

The general architecture presented in the previous section constitutes a preliminary solution to the problem of integrating third-generation knowledge systems and legacy databases. In order to further elaborate and refine this architecture, we have been conducting a series of experiments in the form of actual system implementations. Figure 2 presents a architectural view of one such experiment. In this experiment we examine the processing involved in a particular event in which a legacy knowledge system requests and receives information from a general-purpose database system. Since, this experiment deals with only one knowledge system and only one database, we are able to abstract away a great many of the issues and focus on a specific piece of the problem. The particular issue which this experiment highlights is method-specific transformation.

3.1 General Method

The overall algorithm developed in this experiment breaks down into four steps which correspond to the four architectural components shown in figure 2. These steps are:

1 The *knowledge system* issues a request. This happens when some information is required by the system which is not available in its *local information source.*

2 The *query processor* translates the request into a query. This query is in the language of the *information source.*

3 The *information source* processes the query. It returns some data to the *query processor* which in turn sends the data to the *method specific transformer.*

4 The *method-specific transformer* converts the data into a knowledge representation format which can by understood by the *knowledge system.*

All four of these steps pose complex problems. Executing step one requires that a knowledge system recognize that some element is missing from its knowledge and that that element would help it to solve the current problem; such recognition is a complex though well studied problem. Performing step two requires a mechanism for constructing queries and providing communication to and from the external system; the language and networking issues here are also complex but reasonably well understood. Step three is the fundamental problem of databases: given a query produce a data item; again this is a interesting but thoroughly studied problem. Lastly, step four is can be a challenging problem because the differences between the form of the data in the information source may be arbitrarily complex.

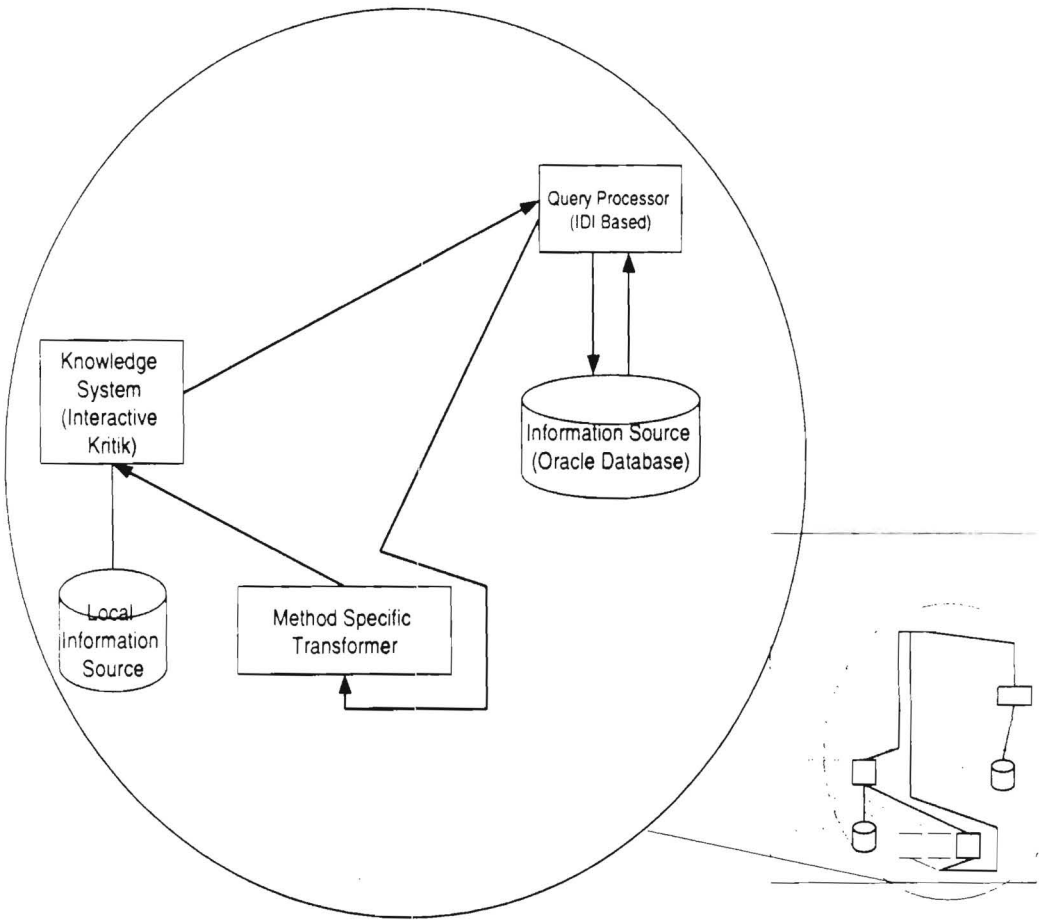


Figure 2: The portion of the architecture relating to the proposed solution

Since all four of these steps are non-trivial, a full description of the algorithm presumably involves an elaboration of each of them. However, the full descriptions of steps one and three are inherently outside of this project, but since we are integrating legacy knowledge systems with legacy database technology it is outside our topic of study to propose algorithms for these components. Step two is clearly an issue within this topic in that query generation and communication are certainly crucial to the integration problem. However, we found that we were able to make use of existing technology for this part of our experimental system: a tool called IDI supports both communication and query generation from within a LISP session. With the first three steps resolved, we were able to focus this experiment on the fourth step: method specific transformation.

By definition, we can not specify a general algorithm for method specific transformation: the algorithm is specific to the reasoning method which uses the knowledge. However we can provide an example of such a transformation and examine how the individual pieces affect the overall effect. The algorithm for the method specific transformer which was implemented in our experimental system is:

4.1 Database data types are coerced into to knowledge system data types.

4.2 Knowledge attributes are constructed from fields in the data item.

4.3 Knowledge attributes are synthesized into a knowledge element.

3.2 Integration

The particular legacy systems which we combined in our implementation were INTERACTIVE KRITIK as the knowledge system and ORACLE as the database system. ORACLE is extremely well known system which uses the relational data model [Codd, 1970], a general purpose knowledge representation formalism which is in extremely common use among database management systems; the interface to ORACLE is the well-known query language, SQL. INTERACTIVE KRITIK is described in detail elsewhere [Goel et al., 1996a, Goel et al., 1996b]; we will only provide a very brief summary here.

INTERACTIVE KRITIK is a knowledge system which performs conceptual design of simple physical devices and provides visual explanations of both the reasoning processes it goes through and the design products it produces. INTERACTIVE KRITIK is an inherently multi-strategy knowledge system. It uses case-based reasoning as a general framework for performing design and it also uses an assortment of model-based methods for doing specific design tasks such as diagnosis and repair.

The experimental code which we have written serves as an interface between INTERACTIVE KRITIK and our ORACLE database. It is used when INTERACTIVE KRITIK is attempting to redesign a device through the use of a component substitution, one redesign strategy in its library of strategies. For example, if the system has determined that a flashlight is not producing enough light, it may decide that a larger bulb is needed. When model-based diagnosis techniques identify a single component whose replacement could potentially solve the design problem, INTERACTIVE KRITIK consults its library of components to see if such a replacement exists; in the example, it would check to see if it knows about a larger bulb and would make a substitution only if it did. In earlier implementations, the library of components was stored entirely within INTERACTIVE KRITIK itself in the form of data structures in memory. In our experiment, these data structures are not present

in memory and the request for an appropriate component takes place through our partial implementation of the pieces of the HIPED architecture illustrated in figure 2.

INTERACTIVE KRITIK sends its request to a query processing module which uses IDI. The request is made as a LISP function call to a function named lookup-database-by-attribute which takes two arguments: a prototype and an attribute value. An example of such a call from the system is a request for a larger light bulb for which the prototype is a symbol 'l-bulb which refers to the general class of light bulbs and a symbol 'capacity-more which is a value for the overall capacity (in lumens) for a light bulb and is internally mapped within INTERACTIVE KRITIK to the number 18. The SQL query generated by IDI for this request is:

```
SELECT DISTINCT RV1.inst_name
FROM   PROTO_INST RV1, INSTANCE RV2
WHERE  RV1.proto_name = 'l-bulb'
AND    RV1.inst_name = RV2.name
AND    RV2.att_val = 'capacity-more'
```

IDI sends this query to the ORACLE database management system running on a remote server. ORACLE searches through a the database tables illustrated in table 1. The first of these tables, INSTANCE, holds the components themselves. The second table, PROTO_INST is a cross-reference table which provides a mapping from components to prototypes.

Table 1: The tables for the ORACLE database

Table INSTANCE		
NAME	ATTRIBUTE	ATT_VAL
littlebulb	lumens	capacity-less
bigmotor	watts	power-more
bigbulb	lumens	capacity-more

Table PROTO_INST	
INST_NAME	PROTO_NAME
littlebulb	l-bulb
bigmotor	motor
bigbulb	l-bulb

If ORACLE finds a result, as it does in this example, it returns it via the method specific transformer. In this case, the query generates the string "bigbulb" as the result; the prototype name and the value are also part of the result but they are not explicitly returned by the database since they were the values used to select the database entry in the first place. The method specific transformer converts the raw data from the database to a form comprehensible to INTERACTIVE KRITIK by using the algorithm described in section 3.1. In step **4.1**, the string "bigbulb" is converted into a LISP symbol. In step **4.2**, the attributes of the new bulb are generated. The symbols 'bigbulb and 'l-bulb are already in the correct form to be the knowledge attributes of name and prototype-comp; the values 'capacity and 'capacity-more are combined into a CLOS object of a class named parameter

and a list containing this one object is created which corresponds to the parameters attribute of the component being constructed. Finally, in step **4.3** these three attribute values are synthesized into a single CLOS object of the component class. This object if then returned to INTERACTIVE KRITIK which is able to continue with its processing.

4 Discussion

The complexity involved in constructing a knowledge system makes reuse an attractive option for true scalability. However, the reuse of legacy systems is non-trivial because we must accommodate the heterogeneity of the "federation" of systems. To require a system to change dramatically to participate in exchanges will, no doubt, limit the scalability of the system by significantly slowing the speed (and willingness of legacy system designers) for integration of a system. The scale comes from the easy integration of legacy systems and transparent access to the resulting pool of legacy knowledge. Sharing data simply requires that a legacy system designer augment the existing schema with metadata that allows a global coordinator to relate data from one system to another, providing a general solution to large scale integration.

The three steps of the method for converting data into knowledge described in Section 3 can all reasonably be considered to be relatively generic units of functionality. It is not unreasonable to suspect that a wide variety of methods might have the same data coercion requirements and thus be able to use the same data coercion routines in their method specific transformers. Furthermore, a great many knowledge systems use representations which are characterized as knowledge elements composed of a set of attribute / value pairs. The general framework for the second and third steps of the algorithm (building attribute / value pairs and then combining them to form a knowledge element) could potentially be applied to a wide variety of knowledge methods; furthermore, to the extent that some methods have similar forms and mechanisms for constructing these elements, they might be able to share specific routines. Our experiment suggests that it may be possible to abstract generic components of method specific transformations; doing so would dramatically mitigate the problem of the quantity of method specific transformers required as the individual transformers could be built from a small, parsimonious set of components. More research is required to fully validate this hypothesis.

The specific experiment described in Section 3 models only a small portion of the general architecture described in Section 2. In a related experiment, we have experimented with another portion of the architecture [Navathe et al., 1996]. In the other experiment, five types of queries that Interactive Kritik may create are expressed in a SQL-like syntax. The queries are evaluated by mapping them using facts about the databases and rules that establish correspondences among data in the databases in terms of relationships such as equivalence, overlap, and set containment. The rules enable query evaluation in multiple ways in which the tokens in a given query may match relation names, attribute names, or values in the underlying databases tables. The query processing is implemented using the CORAL deductive database system [Ramakrishnan et al., 1992].

While the experiment described here demonstrates method-specific transformation of data into knowledge usable by Interactive Kritik, the other experiment shows how queries from the Interactive Kritik can be flexibly evaluated in multiple ways. We expect an integration of the two to provide a seamless and flexible technique for integration of knowledge

systems with databases through method-specific transformation of data into useful knowledge.

Acknowledgements

This work was funded by a DARPA grant monitored by WPAFB, contract #F33615-93-1-1338, and has benefited from feedback from Chuck Sutterwaite of WPAFB. We appreciate the support.

References

[Batini et al., 1986] Batini, C., Lenzernini, M., and Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):325–364.

[Brodie, 1988] Brodie, M. (1988). Reading in artificial intelligence and databases. chapter Future Intelligent Information Systems: AI and Database Technologies Working Together, pages 623–641. Morgan Kauffman.

[Brown and Chandrasekaran, 1989] Brown, D. and Chandrasekaran, B. (1989). *Design Problem Solving: Knowledge Structures and Control Strategies*. Pitman, London, UK.

[Chandrasekaran, 1988] Chandrasekaran, B. (1988). Generic tasks as building blocks for knowledge-based systems: The diagnosis and routine design examples. *Knowledge Engineering Review*, 3(3):183–219.

[Clancey, 1985] Clancey, W. (1985). Heuristic classification. In *Artificial Intelligence*, volume 27, pages 289–350.

[Codd, 1970] Codd, E. (1970). A relational model for large shared data banks. *CACM*, 13(6).

[Finin and Wiederhold, 1991] Finin, T. and Wiederhold, G. (1991). An overview of KQML: A knowledge query and manipulation language. Available through the Stanford University Computer Science Department.

[Genesreth and Fikes, 1991] Genesreth, M. R. and Fikes, R. (1991). *Knowledge Interchange Format Version 2 Reference Manual*. Stanford University Logic Group.

[Goel et al., 1996a] Goel, A., Gomez, A., Grue, N., Murdock, J. W., Recker, M., and Govindaraj, T. (1996a). Explanatory interface in interactive design environments. In Gero, J. S. and Sudweeks, F., editors, *Proc. Fourth International Conference on Artificial Intelligence in Design*, Stanford, California. Kluwer Academic Publishers.

[Goel et al., 1996b] Goel, A., Gomez, A., Grue, N., Murdock, J. W., Recker, M., and Govindaraj, T. (1996b). Towards design learning environments - I: Exploring how devices work. In Frasson, C., Gauthier, G., and Lesgold, A., editors, *Proc. Third International Conference on Intelligent Tutoring Systems*, number 1086 in Lecture Notes in Computer Science, Montreal, Canada. Springer.

[Koch and Loney, 1995] Koch, G. and Loney, K. (1995). *Oracle: The Complete Reference*. Osborne/McGraw Hill/Oracle, 3rd edition.

[Lenat and Guha, 1990] Lenat, D. and Guha, R. (1990). *Building Large Knowledge Based Systems: Representation and Inference in the CYC Project.* Addison-Wesley.

[Marcus et al., 1988] Marcus, S., Stout, J., and McDermott, J. (1988). VT: An expert elevator designer that uses knowledge-based backtracking. In *AI Magazine*, volume 9, pages 95–112.

[McKay et al., 1990] McKay, D., Finin, T., and O'Hare, A. (1990). The intelligent database interface. In *Proc. Eight National Conference on Artificial Intelligence*, pages 677–684, Menlo Park, CA. AAAI.

[Navathe and Donahoo, 1995] Navathe, S. B. and Donahoo, M. J. (1995). Towards intelligent integration of heterogeneous information sources. In *Proceedings of the 6th International Workshop on Database Re-engineering and Interoperability.*

[Navathe et al., 1996] Navathe, S. B., Mahajan, S., , and Omiecinski, E. (1996). Rule based database integration in HIPED: Heterogeneous intelligent processing in engineering design. In *Proc. International Symposium on Cooperative Database Systems for Advanced Applications.* World Scientific Press.

[Paramax, 1993] Paramax (1993). *Software User's Manual for the Cache-Based Intelligent Database Interface of the Intelligent Database Interface.* Paramax Systems Organization, 70 East Swedesford Road, Paoll, PA, 19301. Rev. 2.3.

[Ramakrishnan et al., 1992] Ramakrishnan, R., Srivastava, D., , and Sudarshan, S. (1992). CORAL: Control, relations, and logic. In *Proc. International Conference of the Internation Conference on Very Large Databases.*

[Stroulia and Goel, 1995] Stroulia, E. and Goel, A. (1995). Functional representation and reasoning in reflective systems. *Journal of Applied Intelligence*, 9(1). Special Issue on Functional Reasoning.

Integrating Heterogeneous Databases for Engineering Design *

Sameer Mahajan Shamkant Navathe

College of Computing,
Georgia Institute of Technology,
(sameer.sham)@cc.gatech.edu

Abstract

A number of applications access data residing in heterogeneous databases, based on various data models, having differing schemas, consisting of different internal representations etc. In this paper we pick up a generic application of Engineering Design and assume a predetermined intelligent user interface. We concentrate mainly on relational databases with the SQL interface for the purpose of an illustrative implementation. We demonstrate the use of the CORAL deductive database management system for the representation and maintenance of the metadata repository; and for the generation of multiple possible interpretations of the user queries. CORAL facts store information about the various schemas in the system. CORAL rules establish various relationships amongst different databases, tables, attributes and values. The C++ interface of CORAL (also termed as CORAL++) along with its deductive power is used for arriving at the multiple interpretations of the user queries.

1 Introduction

Heterogeneity of databases is becoming a necessary factor to contend with in the design of new applications because of the proliferation of database management systems that used diverse data models over the last three decades. Among widely implemented data models we have the hierarchical, network, relational and object oriented data models. A large body of work exists that deals with the mapping of these models among one another (e.g. see the mapping of models using the entity relationship model as an intermediate model in [1] [2]. While vendors are also providing middleware solutions to draw data from these legacy systems, the semantic problems of resolving conflicts regarding naming, scale, structure etc. that were pointed out several years ago [3] [4] still remain. The purpose of the project was to tie a set of heterogeneous databases to an "intelligent front end application" which would make requests for data without any knowledge of the schemas of the target databases. To limit the degree of difficulty we assume that we are dealing with data in relational databases only. This assumption is reasonable in the sense that if the data is coming from a hierarchical or a network DBMS, we can first convert the schema to a relational one before treating it for purposes of integration. The database integration problem we discuss here is couched in the context of engineering design which, like any other design application, relies on extracting data from existing databases containing material data, components, existing designs etc. The exact context and the application scenario will be explained in the next section.

We assume that relevant data for the design application is stored in relations (tables) whose schemas are available at "design time". It is conceivable that to support large scale engineering designs, data from a variety of databases, i.e., from multiple schemas would be required. To facilitate integration of data among these databases we assume that the "correspondences", i.e., the similarities and differences among the (meaning of) attributes is encoded in the form of rules. Furthermore, for our application context, the user issues certain queries looking for relevant design information. We

*submitted to Deductive Databases and Logic Programming, Leuven, Belgium 1997

show in this paper how a query may have several interpretations. A deductive database system like CORAL makes it easy to represent the schema information and the interrelationships in a natural way. The C++ interface embedded in CORAL++ makes it easy to write general purpose programs to access and update this information.

2 Problem Definition

Our main objective is to build and demonstrate an intelligent interface to a set of (possibly autonomous) information sources including structured databases, knowledge bases, and unstructured data. The approach we have selected involves the development of a mediator which utilizes meta-knowledge of the underlying information stores to aid a user in browsing data or a system in retrieving specific relevant information.

2.1 Predetermined Querying Interface

The user is assumed to use this system as an Engineering Database for designing purposes. So he would typically like to find components that satisfy his requirements (e.g. batteries with voltage rating higher than 10V and cheaper than $10). Keeping this user's perspective in mind, the Engineering data is thought to be made up of various "Prototypes". Each Prototype has various "Properties". Each Property takes up some "Value" for every Prototype. We can compare the Values of various properties using the relations : ==, <, >, <=, >=, <> etc. For the purpose of our implementation, the queries can be classified into the following five types.

1. (Prototype <proto_name>) : here the user is looking for all the prototypes identified by "proto_name". It is implicit that the user wants to see the various values for various properties of these prototypes.
2. (Property <prop_name>) : the user is interested in all the prototypes having the Property identified by "prop_name". It is implicit that the user wants to see the values taken by this property for the various prototypes, that would be listed.
3. (Prototype <proto_name>) (Property <prop_name>) : the user wants to see all the prototypes identified by "proto_name" and having property identified by "prop_name". It is implicit that the user also wants to see the corresponding value that the property takes for the particular prototype.
4. (Prototype <proto_name>) (Property <prop_name>) (Value <value>) (Relation <rel>) : the user is interested in prototypes identified by "proto_name" having a property identified by "prop_name". In addition to this he wants only those prototypes for which the property takes a value which is related to the given "value" in the query by the relation "rel" (i.e. it is equal to "value" or greater than "value" etc.)
5. (Property <prop_name>) (Value <value>) (Relation <rel>) : the user is interested in all the prototypes for which the property identified by "prop_name" takes a value which is related to the given "value" by the relation "rel".

2.2 Heterogeneous Databases as the Backend

Data is scattered in various databases and various tables in each of those databases. The databases are assumed to be relational with the SQL interface for the illustrative purpose. It is a reasonable assumption and is made to contain the complexity of the problem. The system needs to find out which databases and which tables in these databases have the relevant data to answer a particular query. It then translates the query into a corresponding SQL query for every table. This SQL query is run against that table to get an answer. As we made an assumption of a uniform SQL interface to all the databases, we can simply translate a query into one in SQL against a target database in each of these cases.

3 Overall Operations in the System

The data is organized at two levels namely, (1) the metadata repository : consisting of information about various databases and tables in them and (2) the actual data : which is distributed in various heterogeneous databases. This organization reduces the data to be dealt with at the first level to get to the appropriate database(s) and table(s). It also allows heterogeneity in the various databases involved. The Querying Interface is as described in section 2.1. The "data" together with its "wrapper" forms a database system. "Wrapper" simply defines the access methods to the data for reading and updating purposes. A user query, which is of the form described above, would be translated into the corresponding query, as understood by the corresponding "wrapper", for each of the relevant tables. This query would then be routed to the corresponding database, that contains this table. The metadata repository is consulted in determining these relevant tables and finding the corresponding database. The user would get the result, obtained after running the query against all the applicable databases through the "Result Composer". The overall system architecture is given in figure 1.

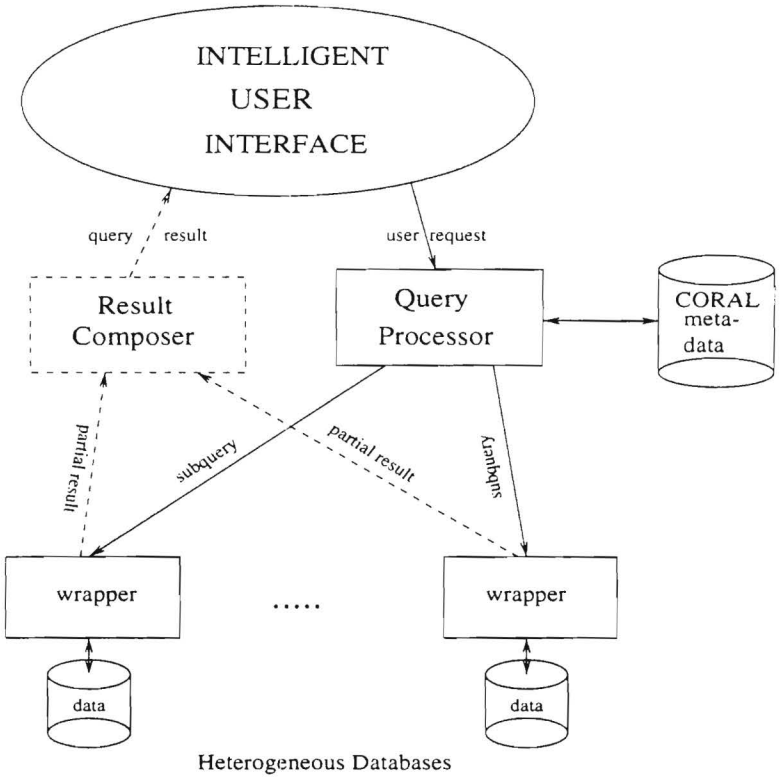


Figure 1: The High Level System View

3.1 Multiple Interpretations of the various types of queries

For better understanding of the following discussion, let us take up an example query. Let the four components of the query be,
(Prototype Battery) (Property Voltage) (Value 10) (Relation ==).
As there can be various tables with different schema, we need to run this query with only those tables that might give meaningful results for the query. We can easily observe that any of "Prototype", "Battery", "Property" and "Voltage" can be a table or a column of a table. The "Battery" and

"Voltage" can also be values in the columns (e.g. those labeled as "Prototype" and "Property" respectively). Of course there are a lot of dependencies amongst these components - e.g., if "Prototype" is a table, then "Battery" has to be a column of this table. On the other hand, if there is a table called the "Battery", then we are looking for values in the column "voltage" or "volts" - so that the query would generate meaningful results with the table. Now we take up an example query for each of the five types listed above. For every query we list the possible tables according to our scheme.

1. (Prototype Battery). The user typically means that he wants all the batteries with their properties and their corresponding values. Hence we will have to run this query against all the tables which,
   - are equivalent to "Prototype Table" and have a column equivalent to "Battery" or
   - are equivalent to "Battery Table"
   - have a column equivalent to "Prototype" (and only the tuples with Prototype as "Battery" would be considered).

   if and only if these tables have columns equivalent to "Property" and "Value" each.
2. (Property Voltage). The user is interested in listing all the Prototypes having "Voltage" as one of their Properties. The Values of these Properties would also be significant from his standpoint. Hence we consider all the tables which,
   - are equivalent to "Property Table" and have a column equivalent to "Voltage" or
   - are equivalent to "Voltage Table"
   - have a column equivalent to "Property" (and only the tuples with Property as "Voltage" would be considered).

   if and only if they have "Prototype" equivalent column.
3. (Prototype Battery) (Property Voltage). The user wants all the batteries with special interest in their voltages. Hence we will run the query against all the tables which,
   - are equivalent to "Prototype Table" and have "Battery", "Property" and "Value" equivalent columns and we would be interested only in the tuples having an entry of "Voltage" in the "Property" equivalent column or
   - are equivalent to "Prototype Table" and have "Battery", "Voltage" equivalent columns or
   - are equivalent to "Battery Table" and have "Property" and "Value" equivalent columns. We would be interested only in those tuples having Property "Voltage" or
   - are equivalent to "Battery Table" and have a column equivalent to "Voltage" Or
   - are equivalent to "Property Table" and have columns equivalent to "Voltage", "Prototype" and "Value". We would be interested in tuples with Prototype as "Battery".
   - are equivalent to "Property Table" and have "Voltage" and "Battery" equivalent columns.
   - are equivalent to "Voltage Tables" and have "Prototype" and "Value" equivalent columns. We would look for only tuples with Prototype as "Battery".
   - are equivalent to "Voltage Table" with "Battery" and "Value" equivalent columns.
   - have "Prototype" and "Property" equivalent columns as far as they have "Value" equivalent column. Only the tuples with Prototype as "Battery" and Property as "Voltage" would be considered.
4. (Prototype Battery) (Property Voltage) (Value 10) (Relation ==). Here the interest is indicated in all the batteries having Voltage as "10". The query can be run with all the tables as indicated as above with an added constraint that only those tuples which have an entry of "10" in the "Voltage" or "Value" column - whichever is applicable - (Note the table can have only one of these columns at a time) will be considered.
5. (Property Voltage) (Value 10) (Relation ==). All the Prototypes having voltage of "10" are being considered. Thus all the tables that,

- are equivalent to "Property Table" and have a column equivalent to "Voltage"
- are equivalent to "Voltage Table" and have a column equivalent to "Value"
- have "Property" and "Value" equivalent columns along with "Prototype" column. (only tuples with Property "Voltage" and Value "10" would be taken into consideration).

would be considered if and only if they have a column equivalent to "Prototype". All the tuples with "Voltage" or "Value" being 10 would be taken into account.

4 Effective Use of CORAL for the Integration

The metadata is stored in the form of CORAL [5] [6] facts and rules. CORAL is a deductive database system which stores data as facts and rules, and allows for that data to be queried. By using CORAL the mediator can decide which database(s) and table(s) are useful in answering any given query. In particular, CORAL is used in deriving relationships like equivalence; between attributes, tables and databases. Any creation, deletion or modification of a table results in a change in the metadata repository. This dynamic behavior can be easily captured by CORAL. In essence, CORAL provides us with the facility for database integration through the facts and rules specified about tables and databases. However, this integration can be considered implicit rather than explicit since no global conceptual schema is explicitly formed. Also the C++ interface provided by CORAL makes writing general purpose programs easy. We explain the implementation by taking help of an example. The integration of CORAL deductive engine with the C++ interface makes CORAL++ rich and well-suited for our system.

4.1 A Simple Example

Consider a query,

```
(Prototype Battery) (Property Voltage).
```

This query is asking for information in all the databases that has something do with battery as a prototype and voltage as a property.

Let us assume a system (as illustrated in figure 2) that has two databases - db1 and db2. Let db1 have the populated tables as shown in Table 1 and Table 2.

CompNo	Prototype	Property	Value
B101	Battery	Voltage	10
M101	Motor	Voltage	10
B110	Battery	Voltage	100
B111	Battery	Current	10
⋮			

Table 1: "Components" Table in db1

and let db2 have populated tables as shown in Table 3 and Table 4.

We observe that according to the discussion in section 3 only the tables in db1 would produce meaningful results with the query under consideration.

4.2 Representation of the Metadata Repository

It is stored as CORAL facts and rules. The advantage of such a storage is that we can utilize the strong deductive power of CORAL (e.g. deducing equivalence of attributes, equivalence of tables etc.). The various components of the repository are described below.

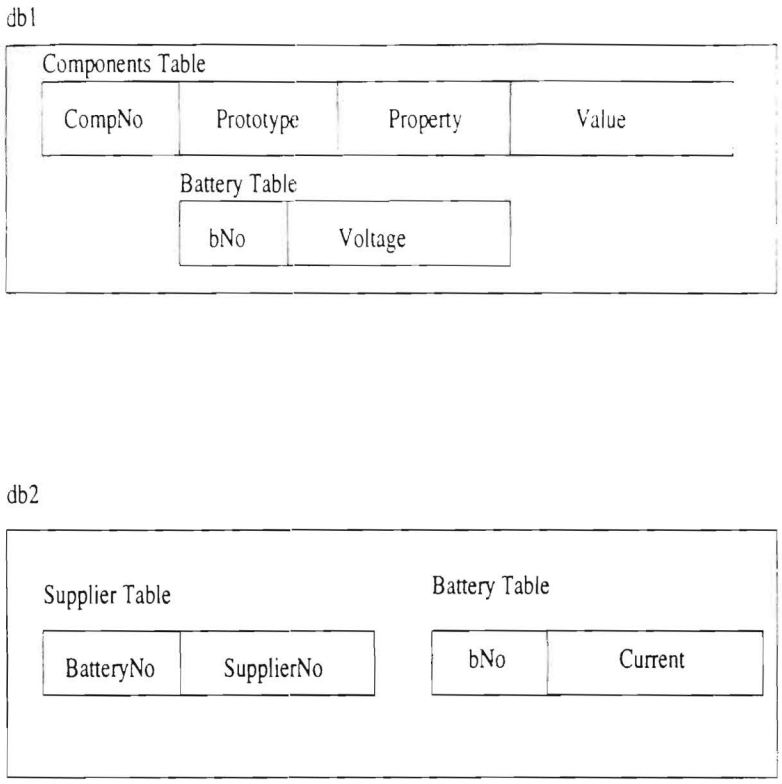


Figure 2: An Example Database System

BatteryNo	Voltage
B101	15
B102	30
⋮	

Table 2: "Battery" Table in db1

BatteryNo	Current
B101	15
B102	30
⋮	

Table 3: "Battery" Table in db2

BatteryNo	Supplier No
B101	4567
B102	4568
⋮	

Table 4: "Supplier" Table in db2

- First we list all the tables in all the databases as CORAL facts like,

```
% For the first database, db1.
belongsTo(component, db1).
belongsTo(battery, db1).

% For the second database, db2.
belongsTo(battery, db2).
belongsTo(supplier, db2).
```

- Then we list attributes of individual tables as CORAL facts. The first argument of these predicates is the database name. It is so because the same table may have different attributes in different databases. e.g. the "battery" table in the two databases "db1" and "db2" as shown below.

```
% for db1
hasAttribs(db1, component, [compName, prototype, property, value]).
hasAttribs(db1, battery, [bName, voltage]).

% for db2
hasAttribs(db2, battery, [bName, current]).
hasAttribs(db2, supplier, [bName, sName]).
```

- We also need facts to list what attributes are equivalent. The equivalence of tables can be either given by facts or can be deduced by the rules (e.g. two tables with equivalent attributes are equivalent). But we do not need them in this particular example.
- To find whether a table has a particular attribute in a given database we define a CORAL rule as,

```
module isAttrib.
export isAttrib(bff).
isAttrib (Db, Table, Attri) :- hasAttribs(Db, Table, Attribs),
                                        iselem (Attri, Attribs).
end_module.

% Module ''iselem'' is defined for the sake of completeness.
module iselem.
export iselem(bb).
@pipelining+.    % Solve in a top-down fashion

iselem(X, [X|_]).
iselem(X, [_|Z]) :- iselem(X,Z).
end_module.
```

4.3 Deducing the Appropriate Tables

It is done according to the scheme suggested in section 3. We use the C++ interface of CORAL for this matter. In fact, an imperative interface (e.g. in C) would have been enough for the purpose. We check for the various conditions given in the scheme and generate the appropriate SQL queries for the existing tables. We run through the algorithm for the example query under consideration,

```
begin
```

```
For every ''table'' equivalent to ''prototype table''
  for every attribute equivalent to ''battery'', say attrib1
    for every attribute equivalent to ''property'', say attrib2
      if ''table'' has ''attrib1'' as well as ''attrib2''
        for every attribute equivalent to ''value'', say attrib3
          if ''table'' has attrib3
            select the corresponding database and fire SQL query,
            SELECT * FROM table
            WHERE attrib2 == voltage equivalent value.
            goto next table
        for every attribute equivalent to ''voltage'', say attrib4
          if ''table'' has attrib4
            select the corresponding database and fire SQL query,
            SELECT * FROM table

For every ''table'' equivalent to ''battery table''
  for every attribute equivalent to ''voltage'', attrib1
    if ''table'' has attrib1
        select the corresponding database and fire SQL query,
        SELECT * FROM table
        goto next table
  for every attribute equivalent to ''property'', attrib2
    for every attribute equivalent to ''value'', attrib3
      if ''table'' has attrib2 and attrib3
        select the corresponding database and fire SQL query,
        SELECT * FROM table
        WHERE attrib2 == voltage equivalent value.

For every table equivalent to ''property table''
  for every attribute equivalent to ''voltage'', say attrib1
    for every attribute equivalent to ''prototype'', say attrib2
      if ''table'' has ''attrib1'' as well as ''attrib2''
        for every attribute equivalent to ''value'', say attrib3
          if ''table'' has attrib3
            select the corresponding database and fire SQL query,
            SELECT * FROM table
            WHERE attrib2 == battery equivalent value.
            goto next table
        for every attribute equivalent to ''battery'', say attrib4
          if ''table'' has attrib4
            select the corresponding database and fire SQL query,
            SELECT * FROM table

For every table equivalent to ''voltage table''
  for every attribute equivalent to ''battery'', attrib1
    if ''table'' has attrib1
      select the corresponding database and fire SQL query,
      SELECT * FROM table
      goto next table
  for every attribute equivalent to ''prototype'', attrib2
    for every attribute equivalent to ''value'', attrib3
      if ''table'' has attrib2 and attrib3
```

```
        select the corresponding database and fire SQL query,
        SELECT * FROM table
        WHERE attrib2 == battery equivalent value.

For every table having columns equivalent to each of
                    prototype, property and value
  select the corresponding database and fire SQL query
  SELECT * FROM table
  WHERE prototype equivalent column) == battery equivalent value
                AND property equivalent column == voltage eqv. value
end
```

4.4 The Result of Sample Queries

Let us say that the wrapper of db1 can handle SQL queries. In that case we first select that database and then simply run a query,

```
SELECT *
FROM component
WHERE prototype = ''battery'' AND property = ''voltage''
```

against the first ("component") table in the database. We take similar actions for the other table in (possibly various) databases. The other query in this case would be,

```
SELECT *
FROM battery
```

again with the same database viz. db1. The result is presented to the user as displayed by the corresponding "wrapper". The task of Result Composer is trivial in this case. It needs to simply display the two tables with appropriate header information (e.g. table names etc.). In general it might be required to merge tables coming from multiple databases. It might also be required to take into consideration the interrelationships amongst various various tables, attributes and values in the same database.

5 Exploiting the Deductive Power of CORAL

The equivalence relationships amongst the various attributes are stored as CORAL facts. The deduced equivalences between tables can be added (and modified as and when required) dynamically for performance benefits. Additional relationships, including semantic ones, between attributes and tables can also be easily captured under this scheme. Sometimes it might be necessary to account for these relationships. They might improve the efficiency of arriving at the result in other cases. A sample system with definition of some additional relationships is given in Appendix A. This example is given for illustrative purposes only. It does not address completeness or efficiency. It means that neither all the facts and rules required are given nor the rules are written to ensure optimal search results.

6 Conclusions and Future Work.

Integration of heterogeneous databases is achieved with respect to an Engineering Data Processing Application. The effective use of deductive database engine integrated with the C++ like interface is illustrated with the help of CORAL++ implementation. CORAL++ makes it easy to represent the information and deduce the outcomes in a natural way.

The system makes an assumption that all the databases involved provide an SQL interface. This condition can be relaxed. In this case we need to generate different queries, as understood by each of the databases involved. Currently only the individual tables are checked to see whether they provide satisfactory data to answer a particular query. But it is possible that two or more tables taken separately do not have enough information to answer a query. At the same time, when taken together (e.g. their join), they provide data to answer the query. Consider that there are two tables - which might be in the same database or in different databases - one with columns "Component Number" and "Prototype". The other with columns "Component Number" and "Voltage". Then neither of them provides enough information for the query
(Prototype Battery) (Property Voltage)
But their equijoin with the additional condition of "Prototype == Battery" for the tuples is of interest to us. The extended solution can exhaustively take care of all such cases. The five types of queries, given in section 2.1, were decided intuitively. They can be modified and / or extended based on a systematic treatment of the user needs.

References

[1] C. Batini, S. Ceri, and S.B. Navathe. **Conceptual Database Design: An Entity Relationship Approach**, Benjamin Cummings, August 1991, 470 pp.

[2] R. Elmasri, S Navathe, **Fundamentals of Database Systems**, 2nd Edition, 1994.

[3] J. Larson, S. B. Navathe, and R. Elmasri A Theory of Attribute Equivalence in Databases with Application to Schema Integration, *IEEE Transactions on Software Engineering*, Vol. 15, No. 4, April 1989.

[4] S.B. Navathe, R. Elmasri and J.A. Larson, Integrating User Views in Database Design, *IEEE Computer*, Vol. 19, No. 1, January 1986, pp. 50-62.

[5] R. Ramakrishnan, D. Srivastava and S. Sudarshan, CORAL: Control, Relations and Logic, Proceedings of the International Conference on Very Large Databases, 1992.

[6] R. Ramakrishnan, D. Srivastava, S. Sudarshan and P. Seshadri, Implementation of the CORAL deductive database system, Proceedings of the ACM SIGMOD Conference on Management of Data, 1993.

Appendix A

```
db(db2).
db(db1).

%
% Attributes in  LDB 1 (University Registration)
%
attrib(db1, professor, profname).
attrib(db1, professor, desc).

attrib(db1, student, name).
attrib(db1, student, id).

attrib(db1, project, id).
attrib(db1, project, projname).
attrib(db1, project, desc).

attrib(db1, course, coursename).
attrib(db1, course, desc).

attrib(db1, manage, id).
attrib(db1, manage, profname).
attrib(db1, manage, projname).

attrib(db1, participate, id).
attrib(db1, participate, projname).
attrib(db1, participate, stdid).

attrib(db1, assigned, id).
attrib(db1, assigned, coursename).
attrib(db1, assigned, profname).

attrib(db1, enroll, id).
attrib(db1, enroll, coursename).
attrib(db1, enroll, stdid).

%
% Attributes in  LDB 2 (University Payroll)
%
attrib(db2, project, id).
attrib(db2, project, fund).

attrib(db2, professor, id).
attrib(db2, professor, name).

attrib(db2, student, name).
attrib(db2, student, id).

attrib(db2, department, id).
attrib(db2, department, name).
```

```
attrib(db2, grapay, code).
attrib(db2, grapay, stdid).
attrib(db2, grapay, projid).

attrib(db2, gtapay, code).
attrib(db2, gtapay, stdid).
attrib(db2, gtapay, deptid).

attrib(db2, deptpay, code).
attrib(db2, deptpay, profname).
attrib(db2, deptpay, deptid).

attrib(db2, projpay, code).
attrib(db2, projpay, profname).
attrib(db2, projpay, projid).
%
% Entities (classes) in Two Databases
%
% Entities in  LDB 1 (University Registration)

class(db1, professor).
class(db1, student).
class(db1, project).
class(db1, course).
class(db1, manage).
class(db1, participate).
class(db1, assigned).
class(db1, enroll).

% Entities in  LDB 2 (University Payroll)

class(db2, professor).
class(db2, student).
class(db2, project).
class(db2, department).
class(db2, grapay).
class(db2, gtapay).
class(db2, projpay).
class(db2, deptpay).

%
% Attributes of an individual class listed
%

has_attribs(db2, professor, [id, name, address, officeno]).
has_attribs(db1, student, [id, name, address]).
has_attribs(db1, project, [id, courseno, desc]).
has_attribs(db1, course, [no, name, desc]).
has_attribs(db1, manage, [id, projid, profno]).
has_attribs(db1, participate, [id, stdid, projid]).
has_attribs(db1, assigned, [id, courseno, profno]).
```

```
has_attribs(db1, enroll, [id, stdid, courseno]).

has_attribs(db2, professor, [id, name]).
has_attribs(db2, student, [id name]).
has_attribs(db2, project, [id, fund]).
has_attribs(db2, department, [id, name]).
has_attribs(db2, grapay,[code, projid, stdid]).
has_attribs(db2, gtapay,[code, deptid, stdid]).
has_attribs(db2, projpay,[code, projid, profid]).
has_attribs(db2, deptpay, [code, deptid, profid]).

% Information from Schema Analysis
%
%    1. Attributes Relationship

keqv(db1, project, id, db2, project, id).
keqv(db1, student, id, db2, student, id).
keqv(db1, student, name, db2, student, name).
keqv(db1, professor, name, db2, professor, name).

kcontain(db2, student, id, db2, grapay, stdid).
kcontain(db2, student, id, db2, gtapay, stdid).

kcontain(db1, student, id, db1, participate, stdid).

kcontain(db2, professor, name, db2, project, name).
kcontain(db2, professor, name, db2, department, name).

kcontain(db1, professor, name, db1, assigned, name).
kcontain(db1, professor, name, db1, manage, name).

kcontain(db1, course, name, db1, assigned, name).
kcontain(db1, course, name, db1, enroll, coursename).

% Information from Schema Analysis
%
%    1. Class  Relationship
% The following relationships would be added dynamically as they are
% deduced by the system.

ksubsume(db1, professor, db2, professor).
ksubsume(db1, project, db2, project).
ksubsume(db1, student, db2, student).

%
% equivalent attributes
%

eqv(X, X) :- attrib(Db, T, X).
eqv(X, Y) :- keqv(Db1, T1, X, Db2, T2, Y).
eqv(X, Y) :- keqv(Db1, T1, Y, Db2, T2, X).
eqv(X, Z) :- eqv(X, Y), eqv(Y, Z).
```

```
%
% Attribute Containment Relationship
%

contain(X, Z) :- eqv(X, Z).
contain(X, Y) :- kcontain(Db1, T1, X, Db2, T2, Y).
contain(X, Z) :- contain(X, Y), contain(Y, Z).
contain(X, Z) :- eqv(X, Y), contain(Y, Z).

%
% Overlapping Attributes
%

overlap(X, Y) :- contain(X, Y).
overlap(X, Y) :- contain(Y, X).
overlap(X, Y) :- eqv(X, Y).

%
% Disjoint Attributes
%

disjoint(X, Y) :- attrib(Db1, T1, X), attrib(Db2, T2, Y), not overlap(X, Y).

%
% equivalent classes
%

eqclass(X, Y) :- keqclass(Db1, X, Db2, Y).
eqclass(X, X) :- class(Db, X).
eqclass(X, Y) :- eqclass(Y, X).
% class equivalence derived from attributes.
eqclass(X, Y) :- has_attribs(X, Z), has_attribs(Y, W), eqlist(Z,W).
eqclass(X, Z) :- eqclass(X, Y), eqclass(Y, Z).

%
% Equivalent Lists of Attributes
%
eqlist([],[]).
eqlist([X|Y],[W|Z]) :- eqv(X,W), !, eqlist(Y,Z).
eqlist([X|Y],[W|Z]) :- iseqv(X,Z),eqlist(Y,[W|Z]).

%
% Testing for being a member of a list.
%
iseqv(X,[]) :- !, fail.
iseqv(X,[Y|_]) :- eqv(X,Y).
iseqv(X,[_|Y]) :- iseqv(X,Y).

%
% class X subsumes class Y
%
```

```
subsume(X, Y) :- ksubsume(Db1, X, Db2, Y).
subsume(X, Z) :- class(Db1, X), class(Db2, Y), class(Db3, Z),
subsume(X, Y), subsume(Y, Z).
subsume(X, Z) :- eqclass(X, Z).
subsume(X, Z) :- eqclass(X, Y), subsume(Y, Z).

%
% overlapping classes
%

overclass(X, Y) :- subsume(X, Y).
overclass(X, Y) :- subsume(Y, X).
overclass(X, Y) :- eqclass(X, Y).

%
% disjoint classes
%

disclass(X, Y) :- class(Db1, X), class(Db2, Y), not overclass(X, Y).
```

PART 2

VISUALIZATION AND USER INTERFACE TECHNIQUES FOR INFORMATION RETRIEVAL

PUBLICATIONS (PART2):

[2.1]. Visual Interface for Textual Information Retrieval Systems", Aravindan Veerasamy, Scott Hudson, Shamkant Navathe. In *Proceedings of IFIP 2.6 3rd Working Conference on Visual Database Systems* 1995, Elsevier, North Holland, pp. 333-345.

[2.2]. ``Querying, Navigating and Visualizing a Digital Library Catalog", Aravindan Veerasamy, Shamkant Navathe. In *Second International Conference on the Theory and Practice of Digital Libraries* ,June 11-13, 1995, Austin, TX

[2.3].``Interactive TREC-4 at Georgia Tech", Aravindan Veerasamy. In *Fourth Text REtrieval Conference*, Oct, 1995, Gaithersberg, MD

[2.4]. ``Evaluation of a tool for visualization of information retrieval results", Aravindan Veerasamy, Nick Belkin. In *Proceedings of the SIGIR 1996, the 19th Annual International Conference on Research and Development in Information Retrieval.*, ACM, New York.

[2.5]. "Effectiveness of a graphical display of retrieval results", Aravindan Veerasamy and Russell Heikes. In *Proceedings of the SIGIR 1996, the 19th Annual International Conference on Research and Development in Information Retrieval.*. ACM, New York.

Visual Interface for Textual Information Retrieval Systems* †

A. Veerasamy, S. Hudson and S. Navathe
College of Computing, 801, Atlantic Drive, Georgia Institute of Technology, Atlanta, Georgia 30332-0280, USA.
Email: {veerasam, hudson, sham}@cc.gatech.edu

Abstract

A prototype user interface implementation for text information retrieval system is described. Using a visualization scheme, the interface provides visual feedback to the user about how the query words influence the ranking of retrieved documents. The interface also helps the user in constructing complex structured queries by simple drag-and-drop operations. An intuitive model where the user classifies the information provided to him/her as being positive and negative aids him/her in supplying rich relevance feedback information to the system. Our prototype interface has been built on top of INQUERY [CCH92]. Preliminary experience with the interface shows it to be a valuable tool in aiding the interactive search process between the user and the system. To test the effectiveness of the interface, we plan to conduct studies on users with real information need searching a large corpus of articles.

Keywords

Visualization of results, visual query languages, query processing, information retrieval

1 User Interface issues for Information Retrieval systems

User Interface issues and interaction techniques for full text information retrieval systems have in general received much less attention than system issues like document representation and retrieval algorithms. We have developed an interface that facilitates the user in visually constructing powerful queries for ranked output retrieval systems. The interface

*This work was supported in part by ARPA Grant No. F33615-93-1-1338 under the Intelligent Integration of Information Program

†Appeared in the Proceedings of the Third IFIP 2.6 Working Conference on Visual Database Systems, 1995

includes a scheme for visualizing the query results in a form that enables the user to see the relationships between the query results and the query. While a majority of online library catalog systems use a boolean model of retrieval, a vast majority of existing experimental information retrieval systems retrieve a ranked set of documents in decreasing order of relevance in response to a free-form textual query. In ranked output systems, the documents and the queries are modeled by a set of weighted index terms. The index term weighting function for the documents primarily takes into consideration

- the frequency of occurrence of the index term in the document,
- the number of documents in the corpus containing that index term.

The effectiveness of a retrieval system is measured by two metrics: recall (the ratio of the number of relevant documents retrieved to the total number of relevant documents in the corpus) and precision (the ratio of the number of relevant documents retrieved to the total number of documents retrieved). The reader is referred to [BC87, Rij79, SM83] for a comprehensive description of evaluation metrics of information retrieval systems, document representation and retrieval techniques.

While processing a free-form textual query, most ranked output Information Retrieval systems automatically extract index terms from the query and weight them. The weighted query index terms are then matched against the weighted index terms of documents to retrieve a ranked set of documents in decreasing order of relevance. Each document is weighted, the higher the weight of a document, the more likely it is to be relevant to the query. Most of the existing library information systems (On-line Public Access Catalogs, OPAC) follow a boolean retrieval model. In this model, the documents retrieved in response to a boolean query are not ranked. If a document satisfies the boolean query specification, it is retrieved. Compared to boolean systems, ranked output systems are a significant improvement since the query can be in a free-form text as opposed to a strict boolean syntax. Also, the retrieved documents are ranked, thereby placing the more useful documents at the top of the list. This is a particularly useful feature since it has been shown that users of boolean systems spend a considerable effort in reducing the size of the result set [Spi93]. On the other hand, ranked output systems introduce a new problem: For a naive user, the logic behind the ranking of documents in response to a query is not as apparent and straightforward as a boolean system. The interface we have developed is aimed at alleviating this problem. It helps the user in understanding how the system computed the ranking of retrieved documents by visualizing the relationship between query terms and the results of the query.

The interface also aids the user in formulating complex structured queries by graphically manipulating objects on the screen. A simple mechanism of classifying any information on the screen into positive and negative instances lends itself to easy formulation of structured queries. The interface is built using Tcl/Tk [Ous94] on top of INQUERY [CCH92], a ranked output retrieval system based on Bayesian inference networks. The interface supports two types of feedback:

- feedback from the user to the system and

- feedback from the system to the user.

It is interesting to note that the term "feedback" in the field of Information Retrieval typically refers to user's feedback to the system, while in the field of Human Computer Interfaces, "feedback" usually refers to the system's feedback to the user. The user's feedback to the system and the different levels of granularity at which the feedback can be provided is discussed in section 4. The system's feedback to the user and the visualization technique is discussed in section 5.

2 Related Work

Numerous studies on user interaction with online library access catalog systems with a boolean retrieval model have been conducted [Spi93, SS92, Dal90, Fid91a, Fid91b, Fid91c]. Spink [Spi93] studies the different forms of user feedback during a retrieval session. Of the total number of feedback actions by the user, 45% were aimed at adjusting the size of the retrieved set of documents, and about 40% were related to relevancy of documents. Fidel [Fid91a, Fid91b, Fid91c] discusses the issue of user interaction by studying the process of search term selection and searching styles in online library access catalogs. Dalrymple [Dal90] looks at the feedback process from a user-centered perspective. Bates [Bat90] describes a boolean retrieval system which integrates an online thesaurus. None of the above studies involve a ranked output system supporting free-form textual queries. All of the systems deal with boolean retrieval model only. We believe that there is a significant difference in the way users interact with a boolean system and a ranked output system. The reader is referred to [Har92] and [HB92] for a comparative discussion of boolean systems and ranked output systems. While building our interface we have borrowed valuable ideas from the studies mentioned above. In particular, the need to integrate an on-line thesaurus with the search interface in an easy-to-use fashion and a simple interaction scheme to include words from documents into the query have been influenced by the results of above-mentioned studies.

Walker and Beaulieu [Wal87, HB92] describe their OKAPI system which is a ranked output retrieval system for library catalogs. Similarly, Fox [FFS+93] describes their MARIAN sysem which is also a ranked output system for library catalogs based on the vector-space model. While OKAPI has facilities for relevance feedback and query expansion using a thesaurus, it largely lacks any means of providing system feedback to the user about how the ranking was computed. The interface we have developed integrates relevance feedback information from the user as well as feedback from the system illustrating the relationship between query results and query words.

A number of visualization schemes for information retrieval systems have also been proposed. The perspective wall [CRM91] describes a visualization scheme which supports browsing of documents. While such a system will not handle qualitative document classifications such as library subject catalogs, it is very useful for visualizing documents based on data which is linear in nature (like date of publication). Other visualization schemes

such as [Kor91, Spo94, HKW94] have facilities for viewing a large document space. But visualizing the document space along more than 3 - 4 dimensions simultaneously becomes very cumbersome using the above systems. Also, most of them do not provide support for querying with relevance feedback and none of them provide support for query expansion using a thesaurus. The visualization scheme in our interface can gracefully handle much higher number of query word dimensions.

2.1 Novelty of our approach

The novelty of our system is in integrating a diverse set of interaction features in a seamless fashion into a single system thereby facilitating the interactive and iterative nature of the information seeking process. The following features are integrated in our system:

- Using a visualization scheme, the interface provides visual feedback to the user about how the query words influence the ranking of retrieved documents.
- By simple drag-and-drop operations of objects on the screen, the interface facilitates a naive end-user in constructing complex structured queries and in providing relevance feedback. This feedback is utilized by the system in a manner described later.
- The interface integrates an online thesaurus which provides words related to the query that can be used by the user to expand the original query.

Belkin and his group's work [BMC93, BMA+91, HB94] on user interfaces for information retrieval systems elucidates the issues in user interface and interaction techniques for full text retrieval systems. Belkin [BMA+91] mentions that "This type of analysis led to another important conclusion, namely that information systems for end users must support a variety of goals and tasks, but through some common interface or seamless access mechanism to a variety of relevant information sources and system functionalities". Our interface takes a step in that direction by integrating different pieces of information with a visualization scheme and simple interaction techniques.

3 Interactive Construction of Queries

Searching a database for information is a highly interactive process with the user constantly refining the query after examining the results of previous iteration until he/she is either satisfied with the results or is frustrated with the process and gives up. In existing information retrieval systems, the interaction proceeds by the user providing feedback on which of the retrieved documents are relevant to his/her information need. The system uses this information to modify the original query resulting in an improved ranking of retrieved documents. It has also been shown by Spink [SS92] that during iterative query reformulation, users tend to expand the query using search terms from various sources

such as a thesaurus, previously retrieved documents and user's background knowledge. Expanding the query with terms from such sources can contribute to retrieval of more relevant documents in the next iteration.

Our interface encourages the interaction between the user and the system by providing the user with simple interaction technique to let him/her supply relevance feedback at different levels of granularity: whole documents, document portions, phrases and individual words. Almost any information appearing on the screen can be used for feedback. This is achieved by simple "drag-and-drop"ping the feedback object into either a "Positive Objects" window colored green or a "Negative Objects" window colored red. This scheme provides a simple abstraction to the user for classifying any type of information without having to worry about what action to take for what type of information. A typical user session along with the response of the interface for every user action is described below using an example (please refer to Figure 1). The database being queried contains a collection of titles, authors and abstracts of thousands of CACM articles.

- The user types in his free form textual query in the query window. In the example shown in figure 1, the query is "image audio and text data compression".

- As every query word is typed in, the system consults an on-line thesaurus and displays words and phrases related to the query word in an adjacent window.

- At any point during the session the user can drag-and-drop any of the related words/phrases into the positive and negative windows. Internally the system expands the query by treating the positive words/phrases as synonyms of the corresponding query word. The negative words/phrases are included in the query with a NOT operator. For example, if for a query word "bank", the phrase "financial institution" is classified as positive and "river bed" is classified as negative, the corresponding internal query would be "#SYNONYM(bank #2[‡](financial institution)) #NOT(#2(river bed))". The end-result of this classification is a possible improvement in the precision measure since documents containing the phrase "river bed" will be weighted lower than other documents, and a possible improvement in the recall measure since documents containing the phrase "financial institution" are also retrieved. The interface facilitates construction of such structured queries by simple drag-and-drop operations. In the example in figure 1, three words related to the query word "compression", namely, "compaction, "shortening" and "condensation" have been classified as positive. Internally the systems treats these three words as synonyms of "compression".

- After the user types in the query, the system evaluates the query and displays the titles of top-ranked documents in the "Query Results" window.

- The user examines the query result. Double-clicking any title with the mouse will bring up the full document.

[‡]#2 is the proximity operator in INQUERY specifying that the words should appear within a distance of 2 within each other

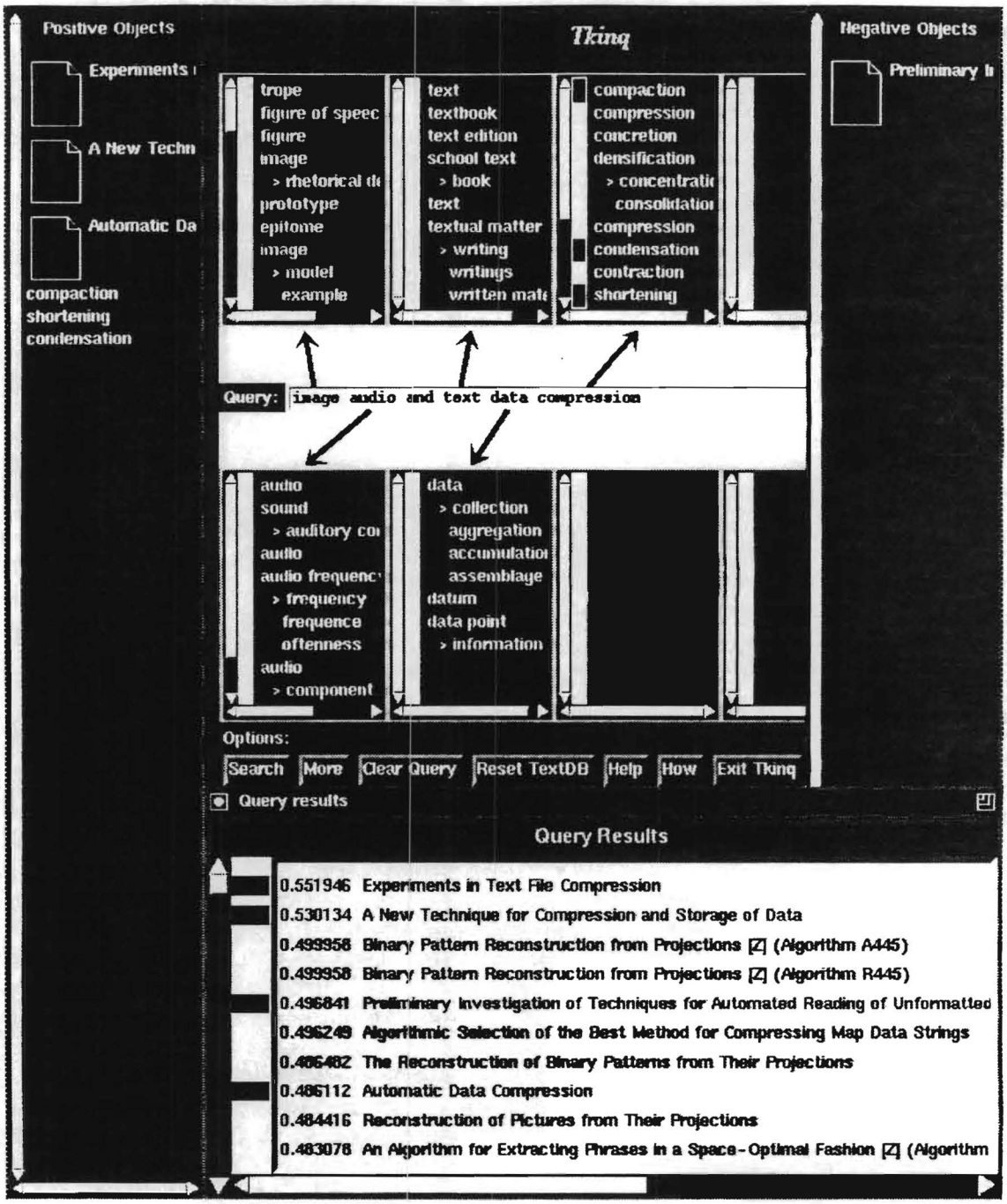


Figure 1: Sample querying session. The window titled "Positive object" is colored green and the window title "Negative Objects" is colored red. When a document is classified as positive/negative, the title of that document in the "Query results" window is also colored green/red.

- The user can classify any document as being relevant or non-relevant by drag-and-drop'ing the document into positive and negative windows. In the example in figure 1, the user has classified three documents titled "Experiments in text file compression", "A new technique for compression and storage of data" and "Automatic data compression" as positive. The document titled "Preliminary investigation of techniques for automated reading of unformatted text" has been classified as negative. Internally, the systems extracts 4 - 6 high frequency words from the positive documents and adds it to the query thereby expanding the query. This results in the retrieval of documents similar to the positive documents.

- The user can also highlight a portion of a document and drag-and-drop it into the positive and negative windows. The words in the highlighted document portion are used to expand the query in the next iteration.

- During the next iteration, the reformulated query with the relevance feedback information is processed by the system resulting in an improved ranking of documents.

The positive and negative windows for feedback are aimed at mimicking the user's view that some information is in line with the information need and some not. After an object has been classified as positive (or negative), the system always colors the object green (or red) whenever the object is displayed, thereby reinforcing the user with the fact that the object is being used for relevance feedback. While arguing for the use of direct manipulation techniques for Information Retrieval, Mitev [Mit89] mentions that

> "Parts of document(s), individual word(s), sentences or groups of word(s) displayed could be used directly as something to be input for another search. This could be done, for example, by pointing and 'picking' them on the screen and carrying them across another area of the screen. The user would not have to input them again."

This is precisely what has been accomplished in our interface. In their retrieval system, Campbell [CS] uses a cut-and-paste mechanism for relevance feedback by letting the user add portions of retrieved documents back into the query window.

This section dealt with the interaction technique to let the user provide relevance feedback information to the system. The next section deals with visual feedback from the system on how the query results were computed.

4 Visualization of query results

While systems with a boolean retrieval model retrieve an unordered set of documents in response to a query, ranked output information retrieval systems retrieve a ranked set of documents. While the reason for retrieving a document is fairly clear in the case of a boolean system, the reason why a document is assigned a specific rank is not apparent

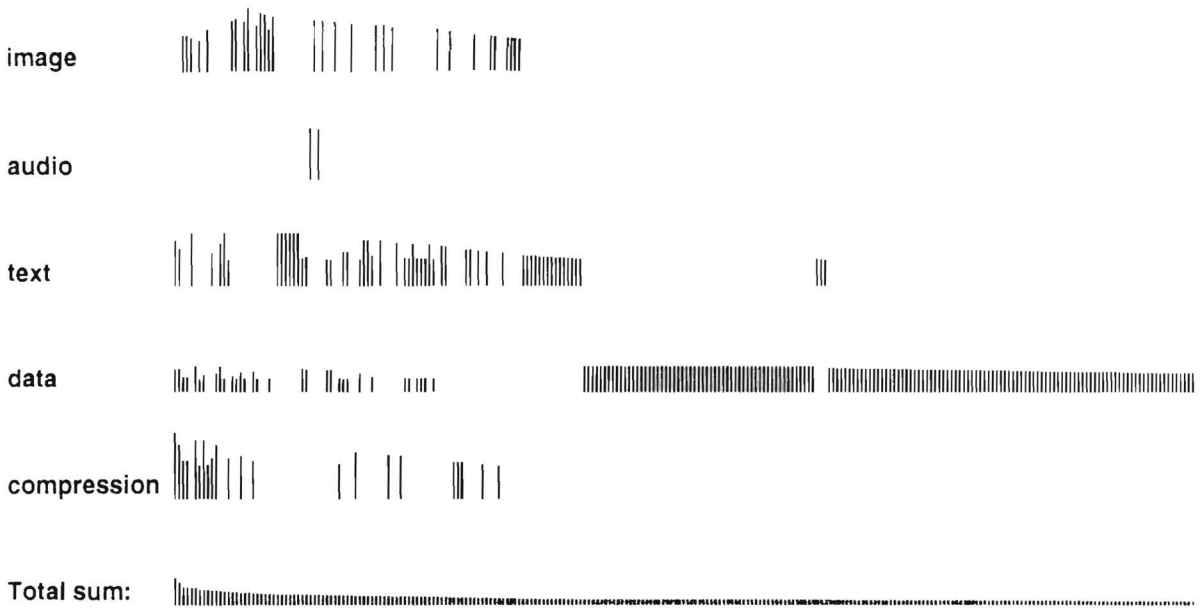


Figure 2: Visualization of results for the base query.

in the case of a ranked output system. Without knowing how the system computed the ranking of documents, the user will have to treat the retrieval mechanism as a black box. We stand to gain a lot by keeping the user more informed about the retrieval process of the system. If the user has more information about how the ranking was computed, he/she will be in a better position to reformulate the query for the next iteration. He/she can take into account the deficiencies of the system in adjusting his/her query. It will also help in reinforcing the right mental model.

In our interface, we keep the user informed about the retrieval mechanism by providing visual feedback about how the query results are related to the query words. This is done by a visualization scheme as shown in the figure 2. The visualization reveals the extent to which each query word was responsible for retrieving the set of documents. The visualization consists of a set of histograms, one for every query word (except stop words) typed in by the user, and one histogram for the total query (labeled "Total sum"). All the histograms are placed one below the other with the "Total sum" histogram appearing at the bottom and the query-word-histograms appearing in the order in which query words were typed in. Each histogram consists of a set of vertical bars, one bar for each retrieved document. For the top ranked document, a vertical bar is drawn in the leftmost position (i.e, lowest X coordinate position) in the "Total sum" histogram. The height of the bar is proportional to the weight of the document. (Note that each document is given a weight. The higher the document weight, the more likely it is to be relevant to the query.) For the same document, vertical bars in the same X-coordinate position are also drawn in the query-word-histograms. The height of the vertical bar in any given query-word-histogram is proportional to the weight of the query word in that document. It represents the contribution of the query word in retrieving that document. If the query word does not

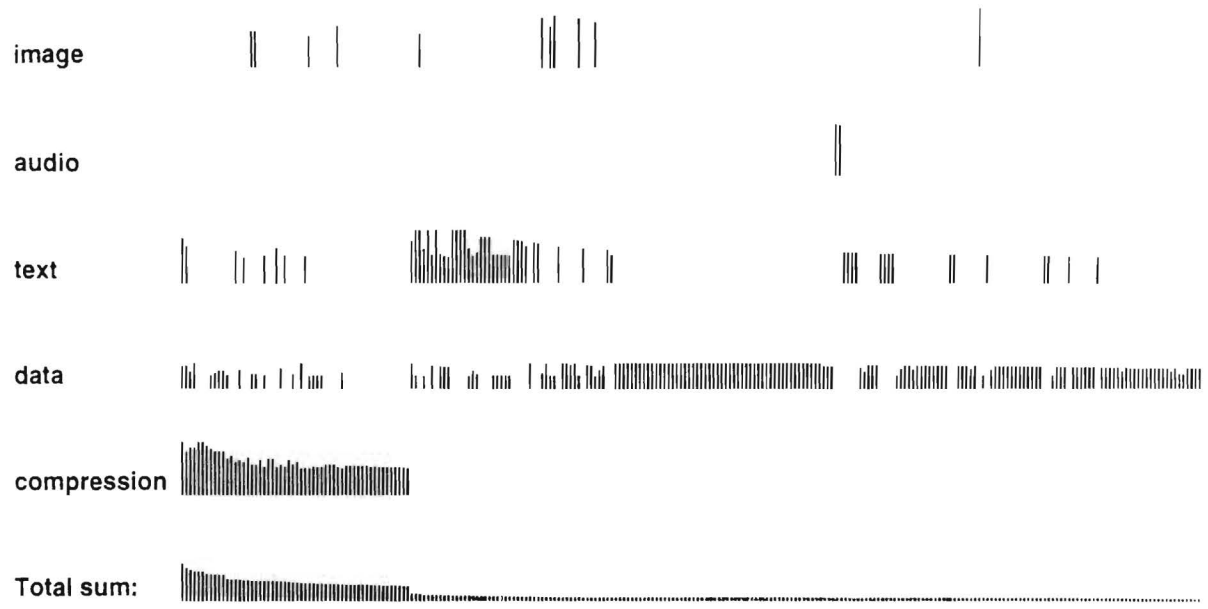


Figure 3: Visualization of results for query with feedback information.

appear in the document, thereby getting a weight of zero, a bar of zero height is drawn which shows up as an empty space in that X-coordinate position. The second ranked document occupies the next higher X-coordinate to the right and so on upto a maximum of top 200 documents.

The visualization shown in Figure 2 corresponds to the base query with no feedback information from the user. We can see that all but two of the top 200 documents have nothing to do with audio. Almost all of the second half of the 200 documents were retrieved because they contained the query word "data". More significantly, only about 10% of the documents have anything to do with compression – which is the crux of the query. This illustrates that the query should be expanded with more words related to "compression". In fact, the decision to classify the three synonyms of "compression" (as shown in figure 1) was made after examining the distribution of "compression" in the visualization. Figure 3 shows the distribution of query terms in the query result for the revised query in the second iteration with all the feedback information. We can see that almost all the documents about "compression" have been ranked at the very top. Also there are more documents retrieved due to "compression" because of the synonyms and the positively classified documents. Our experience with this visualization scheme has shown it to be very useful in identifying differents facets of the query.

5 Conclusion & Future work

A prototype interface for a ranked output information retrieval system has been implemented. The interface facilitates the inherently interactive nature of the information

seeking process. Drag-and-drop operations form the basis of interaction encouraging the user to provide feedback information to the system and helps in the dialog between the user and the system. Almost any information on the screen can be used by the user to provide feedback information. An online thesaurus, WordNet [MBF+90], is integrated with the interface to form a single system.

The interface also supports a visualization scheme which illustrates how the query results are related to the query words. Visualizing the results of the query keeps the user more informed on how the system computed the ranking of documents. With this information, the user is better equipped to reformulate the query for the next iteration. It is our opinion that integrating all of the above features in a seamless interface leads to an interplay between different items that is much more beneficial than the sum of the individual items in isolation.

In demonstrating the system to the reference librarians at Georgia Tech and in observing casual users of the system, we believe that the features we have implemented in this system contributes to enhancing the end-user's interaction with the system. As a result, the system is better able to access the user's need and the user has a better understanding of the system's inference. However we cannot categorically conclude the effectiveness and the utility of the interface without conducting formal user-studies.

In future, we plan to test the effectiveness of the interface by conducting two studies: One with users having real information needs searching a traditional library database and another with volunteers searching the TREC [Har94] document collection with supplied search statements. Since all the relevant documents for the supplied search statements in the TREC collection are known, recall and precision of searches performed with our interface can be compared against other systems.

6 Acknowledgments

We are thankful to Dr. Bruce Croft for letting us use the INQUERY retrieval system. Many thanks to Dr. Marti Hearst whose Tcl/Tk code for the SMART system was helpful as a spring board for us to write the interface.

References

[Bat90] Marcia J. Bates. Design for a subject search interface and online thesaurus for a very large records management database. In *Proceedings of the 53rd Annual Meeting of the American Society for Information Science*, pages 20–28, 1990.

[BC87] Nick Belkin and W.B. Croft. Retrieval techniques. In E. Martha, editor, *Annual Review of Information Science Technology*, pages 110–145. Elsevier Science Publishers, 1987.

[BMA+91] N.J. Belkin, P.G. Marchetti, M. Albrecht, L. Fusco, S. Skogvold, H. Stokke, and G. Troina. User interfaces for information systems. *Journal of Information Science*, 17:327–344, 1991.

[BMC93] N.J. Belkin, P.G. Marchetti, and C. Cool. BRAQUE: Design of an interface to support user interaction in information retrieval. *Information Processing and Management*, 29(3):325–344, 1993.

[CCH92] J.P. Callan, W.B. Croft, and S.M. Harding. The INQUERY retrieval system. In *Third International Conference on Database and Expert Systems Applications*, September 1992.

[CRM91] S. Card, G. Robertson, and J. Mackinlay. The information visualizer, an information workspace. In *Proceedings of CHI 91 Human Factors in Computer Systems.*, 1991.

[CS] I. Campbell and M. Sanderson. Personal communication. University of Glasgow.

[Dal90] P.W. Dalrymple. Retrieval by reformulation in two library catalogs: toward a cognitive model of searching behaviour. *JASIS*, 41(4):272–281, 1990.

[FFS+93] Edward A. Fox, Robert K. France, Eskinder Sahle, Amjad Daoud, and Ben E. Cline. Development of a modern OPAC: From REVTOLC to MARIAN. In Robert Khorfhage, Edie Rasmussen, and Peter Willett, editors, *Proceedings of sixteenth ACM SIGIR Conference*, pages 248–259. ACM SIGIR, June-July 1993.

[Fid91a] Raya Fidel. Searcher's selection of search keys: I. the selection routine. *JASIS*, 42(7):490–500, 1991.

[Fid91b] Raya Fidel. Searcher's selection of search keys: II. controlled vocabulary or free-text searching. *JASIS*, 42(7):501–514, 1991.

[Fid91c] Raya Fidel. Searcher's selection of search keys: III. searching styles. *JASIS*, 42(7):515–527, 1991.

[Har92] Donna Harman. User-friendly systems instead of user-friendly front-ends. *JASIS*, 43(2):164–174, 1992.

[Har94] D.K. Harman, editor. *The Second Text REtrieval Conference (TREC-2)*. NIST Special Publication, March 1994.

[HB92] Micheline Hancock-Beaulieu. User friendliness and human-computer interaction in online library catalogues. *Program*, 26(1):29–37, January 1992.

[HB94] Scott Henninger and Nick Belkin. Tutorial on interface issues and interaction strategies for information retrieval systems. In *Human Factors in Computing Systems CHI 94 Conference Companion*, pages 387–388, 1994.

[HKW94] Matthias Hemmje, Clemens Kunkel, and Alexander Willet. LyberWorld – A visualization user interface supporting full text retrieval. In *Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, pages 249–259, 1994.

[Kor91] Robert Korfhage. To see, or not to see – is that the query? In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 134–141, 1991.

[MBF+90] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: An on-line lexical database. *Journal of Lexicography*, 3(4):235–244, 1990.

[Mit89] Nathalie N. Mitev. Ease of interaction and retrieval in online catalogues: contributions of human-computer interaction research. In Charles R. Hildreth, editor, *The online catalogue*, chapter 8, pages 142–176. Library Association Publishing, London, 1989.

[Ous94] John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[Rij79] Keith Van Rijsbergen. *Information Retrieval*. Butterworths, London, second edition, 1979.

[SM83] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, New York, 1983.

[Spi93] Amanda Spink. Interaction with information retrieval systems: Reflections of feedback. In *Proceedings of the Annual Meeting of the American Society for Information Science*, pages 115–121, 1993.

[Spo94] Anslem Spoerri. InfoCrystal: A visual tool for information retrieval and management. In *Human Factors in Computing Systems CHI 94 Conference Companion*, pages 11–12, 1994.

[SS92] Amanda Spink and Tefko Saracevic. Sources and use of search terms in online searching. In *Proceedings of the 55th Annual Meeting of the American Society for Information Science*, pages 249–255, 1992.

[Wal87] Stephen Walker. OKAPI: Evaluating and enhancing an experimental online catalog. *Library Trends*, Spring:631–645, 1987.

himalaya:veerasam

dl.ps

Tue Apr 29 17:11:24 1997

hobbes / Hobbes

hobbes himalaya:veerasam Job: dl.ps Date: Tue Apr 29 17:11:24 1997

hobbes himalaya:veerasam Job: dl.ps Date: Tue Apr 29 17:11:24 1997

hobbes himalaya:veerasam Job: dl.ps Date: Tue Apr 29 17:11:24 1997

hobbes himalaya:veerasam Job: dl.ps Date: Tue Apr 29 17:11:24 1997

Querying, Navigating and Visualizing a Digital Library Catalog

Aravindan Veerasamy, Shamkant Navathe
College of Computing
801, Atlantic Drive
Georgia Institute of Technology
Atlanta, Georgia 30332-0280, USA.
Phone: 1-404-894-8791
E-mail: {veerasam, sham}@cc.gatech.edu

ABSTRACT

We describe the design of an User Interface for a ranked output Information Retrieval system that integrates querying, navigation and visualization in a seamless fashion. Highlights of the system include the following:

- Using a visualization scheme, the interface provides visual feedback to the user about how the query words influence the ranking of retrieved documents.
- By simple drag-and-drop operations of objects on the screen, the interface facilitates a naive end-user in constructing complex structured queries and in providing relevance feedback.
- To suit the evolving information needs of the user, the interface supports navigational features such as browsing documents by specific authors and browsing the Table of Contents of publications.
- The interface integrates an online thesaurus which provides words related to the query that can be used by the user to expand the original query.

By providing a rich set of features, the interface coherently supports a wide spectrum of information gathering tactics for different classes of users.

KEYWORDS: Visualization of results, visual query languages, query processing, information retrieval

WALK-THROUGH OF A TYPICAL USER SESSION

A typical user session along with the response of the interface for every user action is described below using an example (refer to Figure 1).

- The user types in his/her free form textual query in the query window. In the example shown in figure 1, the query is "ozone depletion and melanoma"
- As every query word is typed in, the system consults an online thesaurus and displays words and phrases related to the query word in an adjacent window.
- At any point during the session the user can "drag-and-drop" (using the mouse) any of the related words/phrases into the positive and negative windows. Internally the system expands the query by treating the positive words/phrases as synonyms of the corresponding query word. The negative words/phrases are included in the query with a NOT operator. For example, if for a query word "bank", the phrase "financial institution" is classified as positive and "river bed" is classified as negative, the corresponding internal query would be "#SYNONYM(bank #2[1](financial institution)) #NOT(#2(river bed))". The interface facilitates construction of such structured queries by simple "drag-and-drop" operations of the mouse. In the example in figure 1, a phrase, namely "skin cancer" that is related to the query word "melanoma" has been classified as positive. Internally the systems treats the phrase as a synonym of "melanoma".
- After the user types in the query, the system evaluates the query and displays the titles of top-ranked documents in the "Query Results" window.
- The user examines the query result. Clicking any title with the mouse will bring up the full document.
- Figure 2 is a visualization of the query results for the base query "ozone depletion and melanoma". The leftmost column of bars corresponds to the top-ranked document, with the columns progressing to the right representing progressively lesser ranked documents. We can see that almost all of the 150 documents were retrieved because they contained the query words "ozone" and "depletion". Only 15 of the top 150 documents have anything to do with melanoma. Further, of those

[1] #2() is the proximity operator in INQUERY specifying that the words inside braces should appear within a distance of 2 of each other in the document.

15 documents, only one discusses ozone (the top-ranked document – leftmost column in Figure 2.) Thus we can clearly see that either there are not many documents dealing with melanoma and ozone or the ozone-layer concept drowns out melanoma during retrieval.

- The user can classify any document as being relevant or non-relevant by "drag-and-drop"ping the document into positive and negative windows. In the example in figure 1, the user has classified two documents titled "CFC-free integral skin foams for steering wheels." and "Video comparator system for early detection of cutaneous malignant melanoma" as positive. The document titled "Symposium on chemistry of the Atmosphere" has been classified as negative.
- The user can also highlight a portion of a document and "drag-and-drop" that portion into the positive and negative windows. The words in the highlighted document portion are used to expand the query in the next iteration.
- During the next iteration, the reformulated query with the feedback information is processed by the system resulting in an improved ranking of documents.
- Figure 3 is a visualization of the results of the revised query (i.e., the query with relevance feedback information). The figure shows that there are four documents dealing with melanoma and ozone. (Note that the documents which deal with melanoma and it's synonym skin cancer are displayed in the same histogram titled "melanoma", since melanoma and skin cancer represent the same query concept). Thus there are three additional documents retrieved due to the effect of classifying the phrase "skin cancer" as a synonym of "melanoma". But still there are not many documents about melanoma compared to ozone depletion. Our experience with this visualization scheme has shown it to be a useful tool for identifying different facets of the query, as in this case, the facets are melanoma and ozone.
- Using any document as a starting point, the user can browse through the list of other articles in the same journal issue or conference proceedings with a help of a Table-of-Contents which is generated automatically. This is useful in many cases such as when the user comes across a special-issue of a journal devoted to the search topic.
- The user can also browse through the list of articles written by the same author. For example, an author who has written an article about the effects of ozone layer depletion on skin cancer has probably authored more articles along the same lines, and the user might want to see them.

CONCLUSION & FUTURE WORK

A prototype interface [4] written in Tcl/Tk [3] using a ranked output information retrieval system, INQUERY [1] for a library catalog, Compendex containing about 300,000 documents has been implemented. The interface facilitates the inherently interactive nature of the information seeking process. "Drag-and-drop" operations (using the mouse) form the basis of interaction encouraging the user to provide feedback information to the system and helps in the dialog between the user and the system. Almost any information on the screen can be used by the user to provide feedback information. An on-line thesaurus, WordNet [2], is integrated with the interface to form a single system.

The interface also supports a visualization scheme which illustrates how the query results are related to the query words. Visualizing the results of the query keeps the user more informed on how the system computed the ranking of documents. With this information, the user is better equipped to reformulate the query for the next iteration. The interface also has facilities to browse the Table of Contents of publications and to browse the list of articles written by a specific author. It is our opinion that integrating all of the above features in a seamless interface leads to an interplay between different items that is much more beneficial than the sum of the individual items in isolation.

We are in the final stages of implementation, and in future, we intend to test the effectiveness of the interface by conducting studies on how library users, experts looking for detailed information as well as naive users, interact with the interface and how they react to ranked output systems as opposed to existing boolean systems. We plan to include a domain-specific thesaurus for the engineering domain from Compendex and a collection-specific word-association thesaurus if possible.

ACKNOWLEDGEMENTS

We are thankful to Dr. Bruce Croft for letting us use the INQUERY retrieval system. We are indebted to the Dean of Georgia Tech Library Ms. Miriam Drake and Engineering Information Inc without whom it would have been impossible to use Compendex data for the experiment. Many thanks to Dr. Marti Hearst whose Tcl/Tk code for the SMART system was helpful as a spring board for us to write the interface. Support in part by ARPA contract No. F33615-93-1-1338 is also appreciated.

REFERENCES

1. J.P. Callan, W.B. Croft, and S.M. Harding. The inquery retrieval system. In *Third International Conference on Database and Expert Systems Applications*, September 1992.

2. George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: An on-line lexical database. *Journal of Lexicography*, 3(4):235–244, 1990.

3. John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

4. A. Veerasamy, S. Navathe, and S. Hudson. Visual interface for textual information retrieval systems. In *To appear in Proceedings of the Third Conference on Visual Database Systems. IFIP 2.6*, 1995.

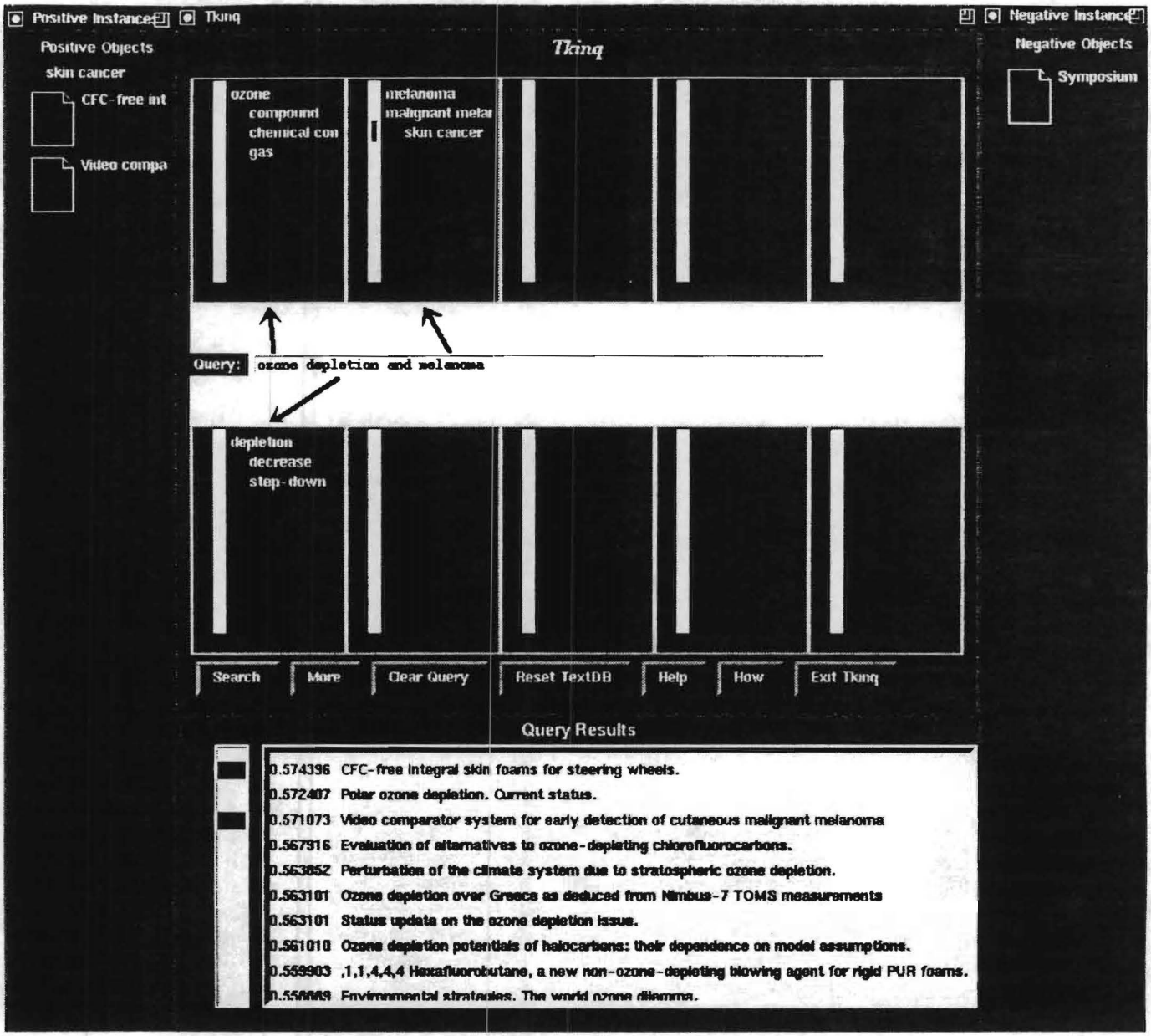


Figure 1: Sample querying session. The window titled "Positive Objects" is colored green and the window titled "Negative Objects" is colored red. All "incantations" of an object in the display are colored green/red whenever it is classified as positive/negative.

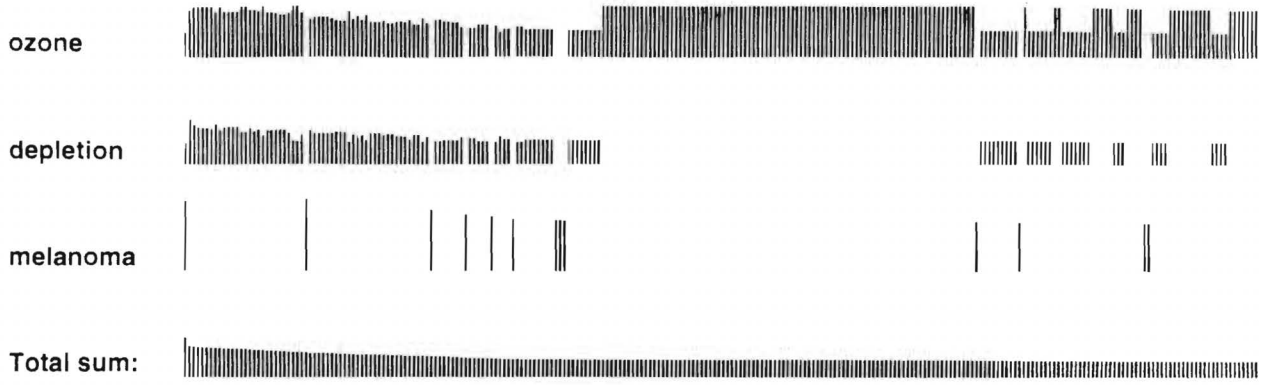


Figure 2: Visualization of results for the base query.

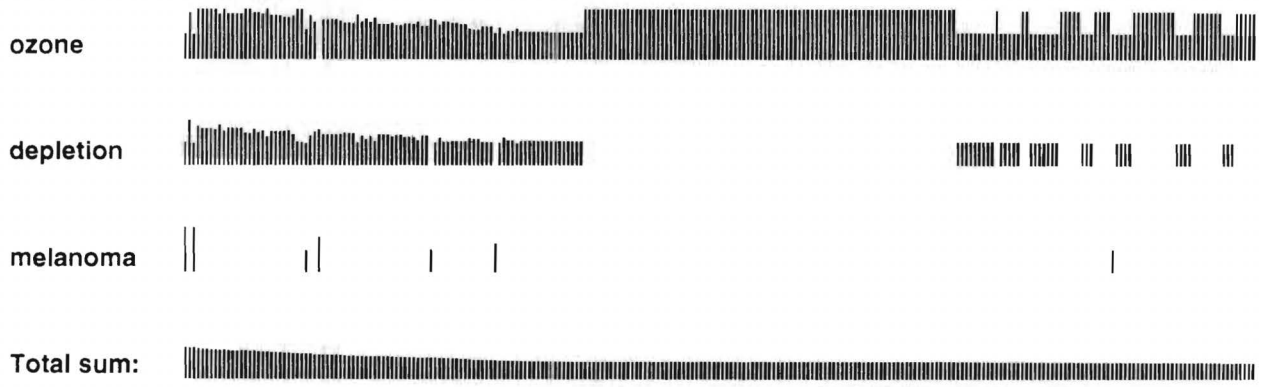


Figure 3: Visualization of results for query with feedback information.

himalaya:veerasam

trec4.ps

Tue Apr 29 17:12:37 1997

hobbes / Hobbes

hobbes himalaya:veerasam Job: trec4.ps Date: Tue Apr 29 17:12:37 1997

hobbes himalaya:veerasam Job: trec4.ps Date: Tue Apr 29 17:12:37 1997

hobbes himalaya:veerasam Job: trec4.ps Date: Tue Apr 29 17:12:37 1997

hobbes himalaya:veerasam Job: trec4.ps Date: Tue Apr 29 17:12:37 1997

Querying, Navigating and Visualizing an Online Library Catalog

Aravindan Veerasamy
Scott Hudson
Shamkant Navathe
{veerasam, hudson, sham}@cc.gatech.edu
College of Computing
801, Atlantic Drive
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
Phone: 404-894-8791
Fax: 404-894-9442

Abstract

We describe the design of an User Interface for a ranked output Information Retrieval system that integrates querying, navigation and visualization in a seamless fashion. Highlights of the system include the following:

- Using a visualization scheme, the interface provides visual feedback to the user about how the query words influence the ranking of retrieved documents.
- By simple drag-and-drop operations of objects on the screen, the interface facilitates a naive end-user in constructing complex structured queries and in providing relevance feedback.
- To suit the evolving information needs of the user, the interface supports navigational features such as browsing documents by specific authors and browsing the Table of Contents of publications.
- The interface integrates an online thesaurus which provides words related to the query that can be used by the user to expand the original query.

By providing a rich set of features, the interface coherently supports a wide spectrum of information gathering tactics for different classes of users.

1 User Interface issues for ranked Information Retrieval systems

User interface issues and interaction techniques for information retrieval have in general received much less attention than system issues like document representation

and retrieval algorithms. It is our belief that the interface should portray a system that is in line with the user's needs and information seeking strategies as opposed to one that only supports querying. We have attempted a holistic approach to building an interface that integrates querying, browsing and visualization all in one system [VNH95].

We present an interaction technique for relevance feedback that can gracefully handle feedback at multiple levels of granularity - whole documents, document portions, phrases and words. The interaction technique tries to mimic the user's view that some of the information displayed by the system is in line with the query and wishes to see more of it, while some information is definitely not what the user intended and should be avoided in future. The user can classify almost any piece of information on the screen as positive or negative by dragging and dropping the information object into either a positive area or a negative area on the screen.

Current experimental ranked output IR systems tend to automate the whole gamut of query processing with tools like natural language processing of queries to identify syntactic constructs and thesaurus to automatically expand query terms with related terms. While such approaches may be successful in future, we believe that the system can be more effective by letting the user provide high quality input such as user selection of thesaurus terms. Voorhees [Voo94] mentions that automatic expansion of query terms using thesaurus words has not been very successful. Spink [SS92] however mentions in her study of source of search terms of real users with intermediaries, that about 20% of the search terms in the final query were from a thesaurus. In an effort to maintain the quality of thesaurus terms used to expand the query, we involve the user in the process of selecting thesaurus terms. This is done by integrating an online thesaurus from which the user picks related words to expand his/her query.

Also, in order to provide high quality feedback information to the system and reformulate the query during subsequent iterations, the user would be in a better position if he/she understands the system and has some idea of how the search results were computed. Seeing a demo of the current interface, reference librarians at our university (who are probably among those most willing and able to formulate the best possible search), were perplexed with the ranking of query result documents and were quite concerned about dealing with a system whose retrieval mechanism for the ranking process is not known. This must be contrasted against the ease with which they can figure out why a set of documents were retrieved in response to a boolean query. So that there is no confusion, we are not implying that the reference librarians prefer boolean queries over free-form queries. On the contrary, they feel that a majority of the users would be more comfortable with free-form queries than with boolean queries. But at the same time, they seem concerned about not knowing

the ranking process "behind the scene". To portray the system as much less of a black box, and to keep the user more informed about how the query result ranking was computed, we use a visualization scheme that shows how the query results are related to the query words.

In order to shape it into a well-rounded IR system, Bates [Bat89] recommends some browsing features that need to be supported. These features include searching the list of references cited by a particular article, searching the list of articles which cite a particular article, browsing the list of articles written by a particular author, browsing all the articles in a particular journal (issue), browsing physically collocated books in an area. Along these lines, we believe that the system should support a rich set of browsing features to enable users with diverse information needs and searching strategies and to help the user through different stages of knowledge acquisition as highlighted by Belkin's notion of ASK [BOB82]. To this end, the interface facilitates browsing articles by authors and browsing the table of contents of journal issues and conference proceedings.

As mentioned above, we address three interface aspects in our system – interaction techniques for relevance feedback which is discussed in section 2, explaining the ranking of documents by means of visualization which is discussed in section 3 and support for browsing in addition to querying which is discussed in section 4.

2 Interactive Construction of Queries and Relevance Feedback

Searching a text database for information is a highly interactive process with the user constantly refining the query after examining the results of previous iteration until he/she is either satisfied with the results or is completely unsuccessful with the process and gives up. In existing information retrieval systems, the interaction proceeds by the user providing feedback on which of the retrieved documents are relevant to his/her information need. The system uses this information to modify the original query resulting in an improved ranking of retrieved documents. It has also been shown by Spink [SS92] that during iterative query reformulation, users tend to expand the query using search terms from various sources such as a thesaurus, previously retrieved documents and user's background knowledge. Expanding the query with terms from such sources can contribute to retrieval of more relevant documents in the next iteration.

Our interface encourages the interaction between the user and the system by providing the user with a simple interaction techniques to let him/her supply rele-

vance feedback at different levels of granularity: whole documents, document portions, phrases and individual words. Almost any information appearing on the screen can be used for feedback. This is achieved by "drag-and-drop"ping of the feedback object into either a "Positive Objects" window colored green or a "Negative Objects" window colored red. This scheme provides a simple abstraction to the user for classifying any type of information without having to worry about what action to take for what type of information. A typical user session along with the response of the interface for every user action is described below using an example (refer to Figure 1).

- The user types in his free form textual query in the query window. In the example shown in figure 1, the query is "ozone depletion and melanoma"
- As every query word is typed in, the system consults an on-line thesaurus and displays words and phrases related to the query word in an adjacent window.
- At any point during the session the user can "drag-and-drop" any of the related words/phrases into the positive and negative windows. Internally the system expands the query by treating the positive words/phrases as synonyms of the corresponding query word. The negative words/phrases are included in the query with a NOT operator. For example, if for a query word "bank", the phrase "financial institution" is classified as positive and "river bed" is classified as negative, the corresponding internal query would be "#SYNONYM(bank #2[1](financial institution)) #NOT(#2(river bed))". The interface facilitates construction of such structured queries by simple "drag-and-drop" operations. In the example in figure 1, one word related to the query word "melanoma, namely, "skin cancer" has been classified as positive. Internally the systems treats the phrase as a synonym of "melanoma".
- After the user types in the query, the system evaluates the query and displays the titles of top-ranked documents in the "Query Results" window.
- The user examines the query result. Double-clicking any title with the mouse will bring up the full document.
- The user can classify any document as being relevant or non-relevant by "drag-and-drop"ping the document into positive and negative windows. In the example in figure 1, the user has classified two documents titled "CFC-free integral skin foams for steering wheels." and "Video comparator system for early detection of cutaneous malignant melanoma" as positive. The document titled "Symposium on chemistry of the Atmosphere" has been classified as negative.

[1] #2() is the proximity operator in INQUERY specifying that the words inside braces should appear within a distance of 2 of each other in the document.

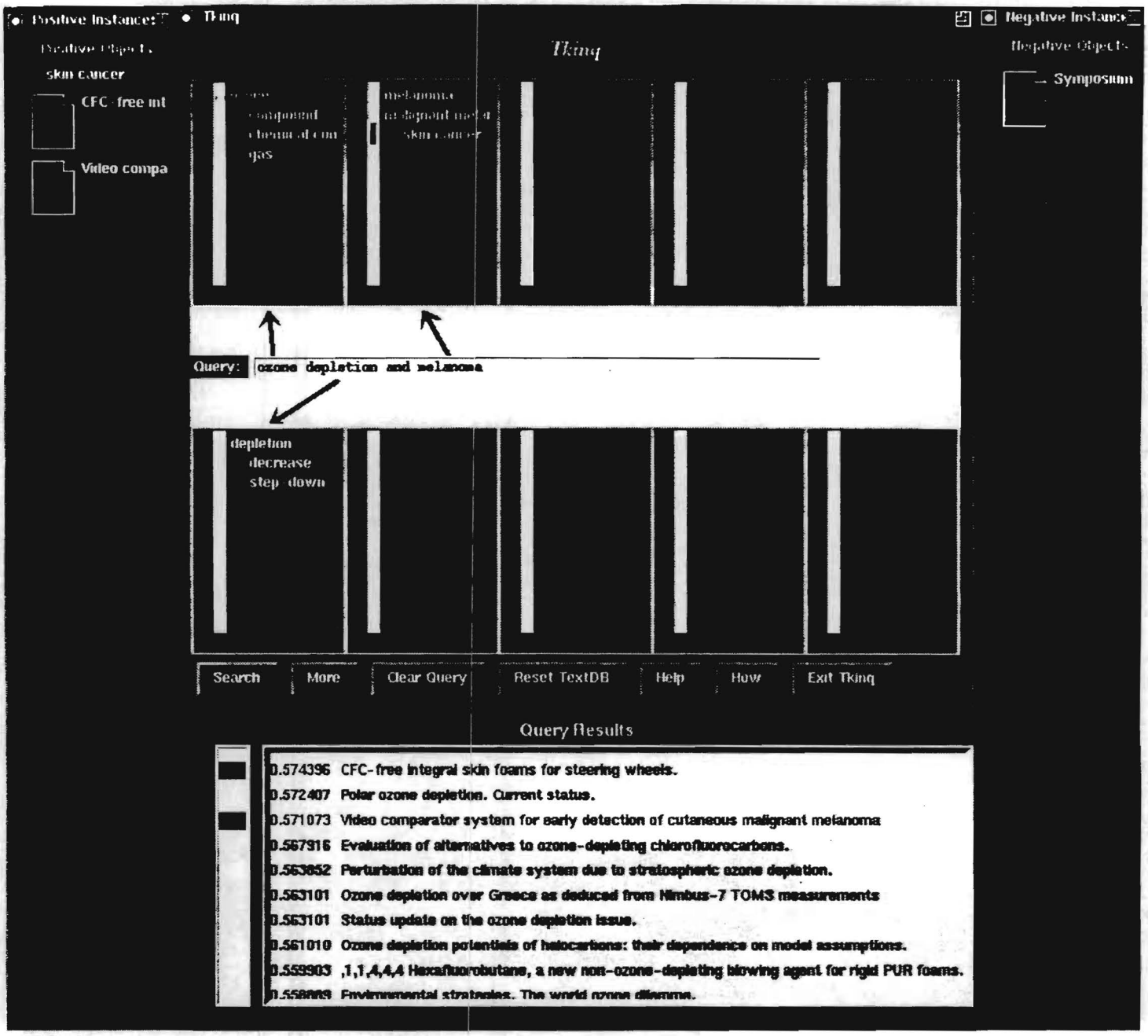


Figure 1: Sample querying session. The window titled "Positive Objects" is colored green and the window titled "Negative Objects" is colored red. When a document is classified as positive/negative, the title of that document in the "Query Results" window is also colored green/red.

- The user can also highlight a portion of a document and "drag-and-drop" that portion into the positive and negative windows. The words in the highlighted document portion are used to expand the query in the next iteration.

- During the next iteration, the reformulated query with the feedback information is processed by the system resulting in an improved ranking of documents.

The positive and negative windows for feedback are aimed at mimicking the user's view that some information is in line with the information need and some not. After an object has been classified as positive (or negative), the system always colors the object green (or red) whenever the object is displayed, thereby reinforcing the user with the fact that the object is being used for relevance feedback. While arguing for the use of direct manipulation techniques for Information Retrieval, Mitev [Mit89] mentions that

> "Parts of document(s), individual word(s), sentences or groups of word(s) displayed could be used directly as something to be input for another search. This could be done, for example, by pointing and 'picking' them on the screen and carrying them across another area of the screen. The user would not have to input them again."

This is precisely what has been accomplished in our interface. In their retrieval system, Campbell [CS] uses a cut-and-paste mechanism for relevance feedback by letting the user add portions of retrieved documents back into the query window.

This section dealt with the interaction technique to let the user provide relevance feedback information to the system. The next section deals with visual feedback from the system on how the query results were computed.

3 Visualization of query results

While systems with a boolean retrieval model retrieve an unordered set of documents in response to a query, ranked output information retrieval systems retrieve a ranked set of documents. While the reason for retrieving a document is fairly clear in the case of a boolean system, the reason why a document is assigned a specific rank is not apparent in the case of a ranked output system. Without knowing how the system computed the ranking of documents, the user will have to treat the retrieval mechanism as a black box. The system stands to gain a lot by keeping the user more informed about the retrieval process of the system. If the user has more information

about how the ranking was computed, he/she will be in a better position to reformulate the query for the next iteration. He/she can take into account the deficiencies of the system in adjusting his/her query. It will also help in reinforcing the right mental model.

In our interface, we keep the user informed about the retrieval mechanism by providing visual feedback about how the query results are related to the query words. This is done by a visualization scheme as shown in the figure 2. The visualization reveals the extent to which each query word was responsible for retrieving the set of documents. The visualization consists of a set of histograms, one for every query word (except stop words) typed in by the user, and one histogram for the total query (labeled "Total sum"). All the histograms are placed one below the other with the "Total sum" histogram appearing at the bottom and the query-word-histograms appearing in the order in which query words were typed in. Each histogram consists of a set of vertical bars, one bar for each retrieved document. For the top ranked document, a vertical bar is drawn in the leftmost position (i.e, lowest X coordinate position) in the "Total sum" histogram. The height of the bar is proportional to the weight of the document. (Note that each document is given a weight. The higher the document weight, the more likely it is to be relevant to the query.) For the same document, vertical bars in the same X-coordinate position are also drawn in the query-word-histograms. The height of the vertical bar in any given query-word-histogram is proportional to the weight of the query word in that document. It represents the contribution of the query word in retrieving that document. If the query word does not appear in the document, thereby getting a weight of zero, a bar of zero height is drawn which shows up as an empty space in that X-coordinate position. The second ranked document occupies the next higher X-coordinate to the right and so on upto a maximum of top 200 documents.

The visualization shown in Figure 2 corresponds to the base query with no feedback information from the user. We can see that only fifteen of the top 200 documents have anything to do with melanoma. Almost all of the 200 documents were retrieved because they contained the query words "ozone" and "depletion". Further, of those fifteen documents, only one discusses ozone (the top ranked document – leftmost bar in Figure 2.) Thus we can see clearly that there are not many documents that discuss the the base query about the effects of ozone layer depletion on melanoma. Either there are not many documents about the effects of ozone layer on melanoma or the concept of ozone layer drowns out melanoma. The visualization scenario after providing feedback (that "skin cancer" is a synonym of "melanoma") and computing the results is shown is shown in Figure 3. The figure shows that there are four documents dealing with melanoma and ozone. (Note that the documents which deal with melanoma and it's synonym skin cancer are displayed in the same histogram titled

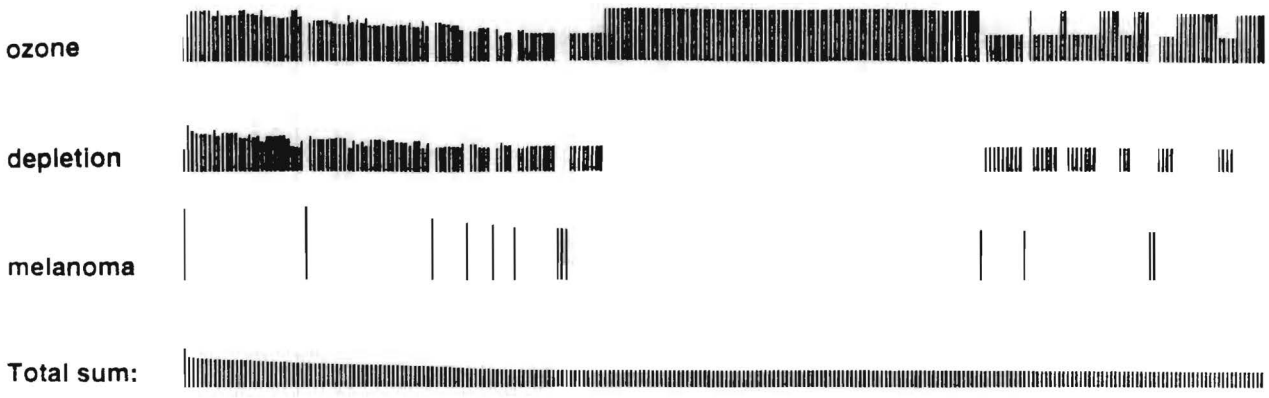


Figure 2: Visualization of results for the base query.

"melanoma", since melanoma and skin cancer represent the same query concept). Thus there are three additional documents retrieved due to the effect of classifying the phrase "skin cancer" as a synonym of "melanoma". But still there are not many documents about melanoma compared to ozone depletion. Our experience with this visualization scheme has shown it to be a useful tool for identifying different facets of the query, as in this case, the facets are melanoma and ozone. The visualization also illustrates which of the query words play a dominant part in retrieving the results and the user has a better idea of what type of query modification is necessary during the next iteration.

4 Browsing

The motivation to integrate browsing features in a querying system has been strongly influenced by [Bat89], [Hil89] and [BOB82]. While it has been argued by all of them that browsing is a central information seeking strategy commonly employed by users, we do not know of any existing online library catalog that integrates browsing and querying. Some examples of browsing activity performed by researchers are:

- While coming across a special issue of a journal or a conference publication devoted to the researcher's are of interest, he/she browses through the table of contents and some articles in the publication.
- On identifying a journal specific to the researcher's are of interest, he/she would want to browse through the publication to keep up-to-date on the developments in the field [Bat89].

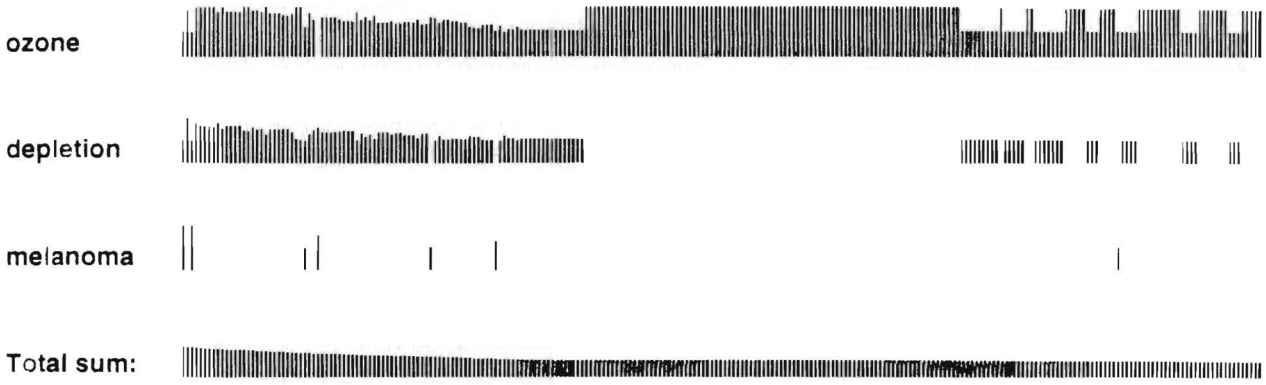


Figure 3: Visualization of results for query with feedback information.

- On discovering that a particular author is working in the same or closely related area, one might want to browse through articles written by that author [Bat89].

In all these cases, the user need not necessarily have a specific information need in mind and may not be able to formulate a query. Conversely, the user might want to browse through the documents as mentioned above while perusing the results of a previously constructed query. While it is true that a user can get the articles written by an author with an "author search" or the articles in a journal with a "journal search", the user may not necessarily want to abandon the results of the current search to initiate a new search author or journal search. The context switch need to initiate a new search can be distracting and disorienting. Browsing can accomplish the "author search" or "journal search" while maintaining the context. Browsing in our system is illustrated in figures 4 and 5. As shown in figure 4, a document display consists of the title of the documents, its authors, pages, name of the journal/conference proceedings, volume, number, part of the publication and the abstract. The portion of the document which can be clicked upon to browse through related information is underlined. As shown in Figure 4, the journal name and the authors are underlined, and hence browsable. Double-clicking an author or journal name from the document display would initiate an internal search in the system, but externally appears as navigation to the user. In our example, double-clicking the

journal citation "Wave motion VOL 13 NUM 2" displays the table of contents of the that journal issue. The table of contents is shown in Figure 5. Double-clicking any article from the table of contents would display the whole article. Author browsing is supported in a similar way.

5 Related Work

Numerous studies on user interaction with online library access catalog systems with a boolean retrieval model have been conducted [Spi93, SS92, Dal90, Fid91a, Fid91b, Fid91c]. Spink [Spi93] studies the different forms of user feedback during a retrieval session. In her study, Spink [Spi93] mentions that of the total number of feedback actions by the user, 45% were aimed at adjusting the size of the retrieved set of documents, and about 40% were related to relevancy of documents. Fidel [Fid91a, Fid91b, Fid91c] discusses the issue of user interaction by studying the process of search term selection and searching styles in online library access catalogs. Dalrymple [Dal90] looks at the feedback process from a user-centered perspective. Bates [Bat90] describes a boolean retrieval system which integrates an online thesaurus. None of the above studies involve a ranked output system supporting free-form textual queries. All of the above systems deal with boolean systems only. We believe that there is a significant difference in the way users interact with a boolean system and a ranked output system. The reader is referred to [Har92] and [HB92] for a comparative discussion of boolean systems and ranked output systems. While building our interface we have borrowed valuable ideas from the studies mentioned above. In particular, the need to integrate an on-line thesaurus with the search interface in an easy-to-use fashion and a simple interaction scheme to include words from documents into the query have been influenced by the results of above-mentioned studies. We expect that the studies we plan to conduct with library users using our interface will provide important insights into the ways users react to ranked output systems. It is also expected to give us an idea of the set of the most useful features to be supported by a ranked output information retrieval system.

Walker [Wal87, HB92] describes their OKAPI system which is a ranked output retrieval system for library catalogs. Similarly, Fox [FFS+93] describes their MARIAN system which is also a ranked output system for library catalogs based on vector-space model. While OKAPI and MARIAN have facilities for relevance feedback and query expansion using a thesaurus, they largely lack any means of providing system feedback to the user about how the ranking was computed. The interface we have developed integrates relevance feedback information from the user as well as feedback from the system illustrating the relationship between query results and query words.

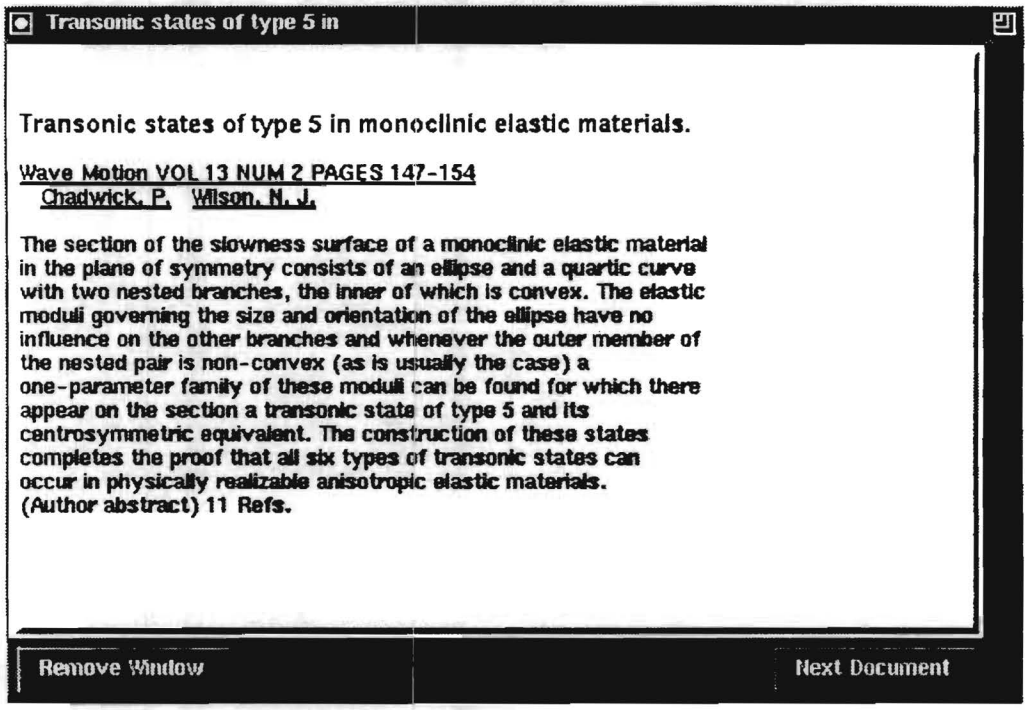


Figure 4: Sample document display. The browsable portions are underlined.

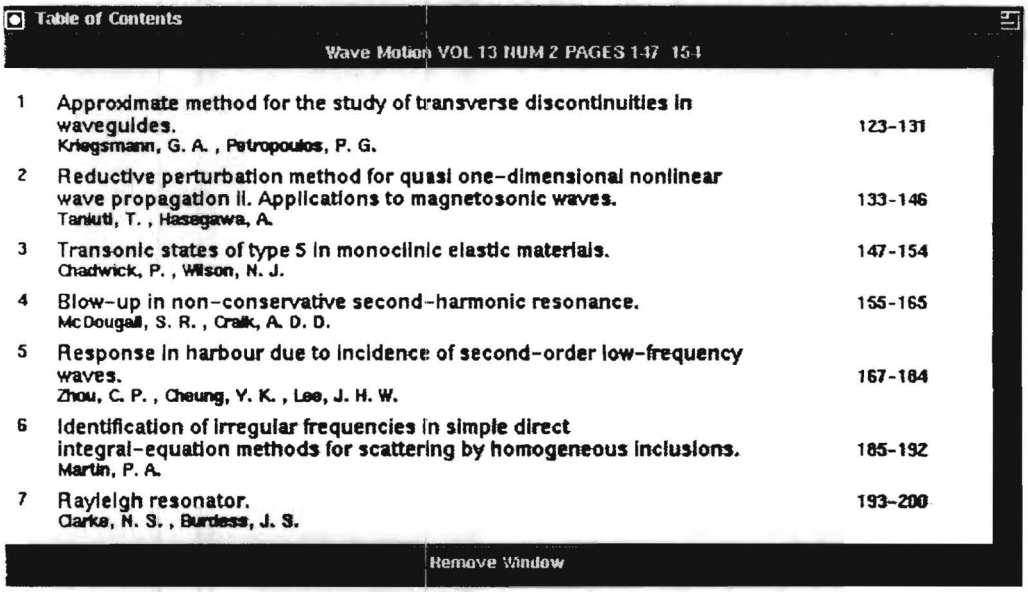


Figure 5: Sample display of table of contents.

A number of visualization schemes for information retrieval systems have also been proposed. The perspective wall [CRM91] describes a visualization scheme which supports browsing of documents. While such a system can not handle qualitative document classifications such as library subject catalogs, it is very useful for visualizing documents based on data which is linear in nature (like date of publication). Other visualization schemes such as [Kor91, Spo94, HKW94] have facilities for viewing a large document space. But visualizing the document space along more than 3 - 4 dimensions simultaneously becomes very cumbersome using their systems. Also, most of them do not support querying with relevance feedback and none of them support query expansion using a thesaurus. The visualization scheme in our interface can gracefully handle much higher number of query word dimensions.

The TileBars work by Marti Heart [Hea95] visually shows the query term ditribution and overlap in retrieved documents. The term distribution in retrieved documents is shown right besides the title of the document. In a number of respects, the reasons and motivations for her work are similar to those of our visualization work [VNH95, VN95]. That both of us seem to have similar motivations behind our work independently of each other reflects on the need for such work. It would be a very interesting exercise to evaluate both TileBars and our visualization work and study their effectiveness in end-user experiments. We are currently undertaking end-user evaluation of our work as part of the interactive track of TREC-4 [TRE95].

Belkin and his group's work [BMC93, BMA+91, HB94] on user interfaces for information retrieval systems elucidates the issues in user interface and interaction techniques for full text retrieval systems. Belkin [BMA+91] mentions that

> ... analysis led to another important conclusion, namely that information systems for end users must support a variety of goals and tasks, but through some common interface or seamless access mechanism to a variety of relevant information sources and system functionalities.

Our interface is a step in that direction by integrating different pieces of information with a visualization scheme and simple interaction techniques.

6 Conclusion & Future work

A prototype interface written in Tcl/Tk [Ous94] using a ranked output information retrieval system, INQUERY [CCH92] for a library catalog, Compendex containing about 300,000 documents has been implemented. The interface facilitates the inherently interactive nature of the information seeking process. "Drag-and-drop"

operations form the basis of interaction encouraging the user to provide feedback information to the system and helps in the dialog between the user and the system. Almost any information on the screen can be used by the user to provide feedback information. An online thesaurus, WordNet [MBF+90], is integrated with the interface to form a single system.

The interface also supports a visualization scheme which illustrates how the query results are related to the query words. Visualizing the results of the query keeps the user more informed on how the system computed the ranking of documents. With this information, the user is better equipped to reformulate the query for the next iteration. The interface also has facilities to browse the Table of Contents of publications and to browse the list of articles written by a specific author. It is our opinion that integrating all of the above features in a seamless interface leads to an interplay between different items that is much more beneficial than the sum of the individual items in isolation.

We are in the final stages of implementation, and in future, we intend to test the effectiveness of the interface by conducting studies on how library users interact with the interface and how they react to ranked output systems as opposed to existing boolean systems. We plan to include a domain-specific thesaurus from Compendex and a collection-specific word-association thesaurus if possible.

7 Acknowledgments

We deeply appreciate the help of Dr. Scott Hudson who was instrumental in his guidance at every stage of the interface development. We are thankful to Dr. Bruce Croft for letting us use the INQUERY retrieval system. We are indebted to the Dean of Georgia Tech Library Ms. Miriam Drake and Engineering Information Inc without whom it would have been impossible to use Compendex data for the experiment. Many thanks to Dr. Marti Hearst whose Tcl/Tk code for SMART was a jumping board to build our system. Support from the ARPA contract No. F33615-93-1-1338 is appreciated.

References

[Bat89] Marcia J. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online Review*, 13(5):407, 1989.

[Bat90] Marcia J. Bates. Design for a subject search interface and online thesaurus for a very large records management database. In *Proceedings of the 53rd Annual Meeting of the American Society for Information Science*, pages 20–28, 1990.

[BMA+91] N.J. Belkin, P.G. Marchetti, M. Albrecht, L. Fusco, S. Skogvold, H. Stokke, and G. Troina. User interfaces for information systems. *Journal of Information Science*, 17:327–344, 1991.

[BMC93] N.J. Belkin, P.G. Marchetti, and C. Cool. Braque: Design of an interface to support user interaction in information retrieval. *Information Processing and Management*, 29(3):325–344, 1993.

[BOB82] N.J. Belkin, R.N. Oddy, and H.M. Brooks. Ask for information retrieval: Parts I and II. *Journal of Documentation*, 38(2,3), 1982.

[CCH92] J.P. Callan, W.B. Croft, and S.M. Harding. The inquery retrieval system. In *Third International Conference on Database and Expert Systems Applications*, September 1992.

[CRM91] S. Card, G. Robertson, and J. Mackinlay. The information visualizer, an information workspace. In *Proceedings of CHI 91 Human Factors in Computer Systems.*, 1991.

[CS] I. Campbell and M. Sanderson. Personal communication. University of Glasgow.

[Dal90] P.W. Dalrymple. Retrieval by reformulation in two library catalogs: toward a cognitive model of searching behaviour. *Journal of the American Society for Information Science*, 41(4):272–281, 1990.

[FFS+93] Edward A. Fox, Robert K. France, Eskinder Sahle, Amjad Daoud, and Ben E. Cline. Development of a modern OPAC: From REVTOLC to MARIAN. In Robert Khorfhage, Edie Rasmussen, and Peter Willett, editors, *Proceedings of sixteenth ACM SIGIR conference*, pages 248–259. ACM SIGIR, June-July 1993.

[Fid91a] Raya Fidel. Searcher's selection of search keys: I. the selection routine. *Journal of the American Society for Information Science*, 42(7):490–500, 1991.

[Fid91b] Raya Fidel. Searcher's selection of search keys: II. controlled vocabulary or free-text searching. *Journal of the American Society for Information Science*, 42(7):501–514, 1991.

[Fid91c] Raya Fidel. Searcher's selection of search keys: III. searching styles. *Journal of the American Society for Information Science*, 42(7):515–527, 1991.

[Har92] Donna Harman. User-friendly systems instead of user-friendly front-ends. *Journal of American Society for Information Science*, 43(2):164–174, 1992.

[HB92] Micheline Hancock-Beaulieu. User friendliness and human-computer interaction in online library catalogues. *Program*, 26(1):29–37, January 1992.

[HB94] Scott Henninger and Nick Belkin. Tutorial on interface issues and interaction strategies for information retrieval systems. In *Human Factors in Computing Systems CHI 94 Conference Companion*, pages 387–388, 1994.

[Hea95] Marti A. Hearst. Tilebars: Visualization of term distribution information in full text information access. In *Proceedings of CHI 95, Denver, Colarado.*, 1995.

[Hil89] Charles R. Hildreth. *The online catalogue*. The library association, 1989.

[HKW94] Matthias Hemmje, Clemens Kunkel, and Alexander Willet. Lyberworld – a visualization user interface supporting full text retrieval. In *Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, pages 249–259, 1994.

[Kor91] Robert Korfhage. To see, or not to see – is that the query? In *Proceedings of the 14th Annual International ACM/SIGIR conference on Research and Development in Information Retrieval*, pages 134–141, 1991.

[MBF+90] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: An on-line lexical database. *Journal of Lexicography*, 3(4):235–244, 1990.

[Mit89] Nathalie N. Mitev. Ease of interaction and retrieval in online catalogues: contributions of human-computer interaction research. In Charles R. Hildreth, editor, *The online catalogue*, chapter 8, pages 142–176. Library Association Publishing, London, 1989.

[Ous94] John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[Spi93] Amanda Spink. Interaction with information retrieval systems: Reflections of feedback. In *Proceedings of the Annual Meeting of the American Society for Information Science*, pages 115–121, 1993.

[Spo94] Anslem Spoerri. Infocrystal: A visual tool for information retrieval and management. In *Human Factors in Computing Systems CHI 94 Conference Companion*, pages 11–12, 1994.

[SS92] Amanda Spink and Tefko Saracevic. Sources and use of search terms in online searching. In *Proceedings of the 55th Annual Meeting of the American Society for Information Science*, pages 249–255, 1992.

[TRE95] In D.K. Harman, editor, *The Third Text REtrieval Conference (TREC-3)*. NIST Special Publication, March 1995.

[VN95] A. Veerasamy and S. Navathe. Querying, navigating and visualizing a digital library catalog. In *Proceedings of the Second International Conference on the Theory and Practice of Digital Libraries*, 1995.

[VNH95] A. Veerasamy, S. Navathe, and S. Hudson. Visual interface for textual information retrieval systems. In *Proceedings of the Third Conference on Visual Database Systems. IFIP 2.6*, 1995.

[Voo94] Ellen M. Voorhees. On expanding query vectors with lexically related words. In Donna K. Harman, editor, *The second Text REtrieval Conference*, pages 223–231, 1994.

[Wal87] Stephen Walker. Okapi: Evaluating and enhancing an experimental online catalog. *Library Trends*, Spring:631–645, 1987.

Interactive TREC-4 at Georgia Tech

Aravindan Veerasamy
veerasam@cc.gatech.edu
College of Computing
801, Atlantic Drive
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
Phone: 404-894-8791
Fax: 404-894-9442

Abstract

At Georgia Tech, we investigated the effectiveness of a visualization scheme for Information Retrieval systems. Displayed like a bar-graph, the visualization tool shows the distribution of query words in the set of documents retrieved in response to a query. We found that end-users use the visualization for two purposes:

- to gain specific information about individual documents – such as the distribution of different query words in that document.
- to gain aggregate information about the query result in general – such as getting a sense of the direction of the query results.

In general they used the visualization tool as much as the title and full text in the process of deciding if a document addresses the given search topic. In structured post-session interviews with searchers, we also obtained information about what the searcher liked, what was frustrating to them, and what they wanted in the system.

1 Introduction

At the TREC-4 interactive experiments at Georgia Tech, we were interested in investigating the effectiveness of a visualization scheme for IR systems that we have developed. The visualization scheme, as given in Figure 2, is intended to provide more information to the user about the query results in addition to just the title and full text. In ranked output systems, the naive end-user has little knowledge about why the system retrieved and ranked the documents in a given way in response to a free-form text query. This problem does not arise in boolean systems since there is no element of surprise in why a particular document was retrieved. The above-mentioned lack of knowledge in ranked output systems can be quite disturbing when a user is not able to get the set of documents he/she needs and does not know enough about the system

to modify the query to get the documents he/she needs. It is with this in mind that we have developed a visualization scheme that shows the distribution of query words in the retrieved documents. This visual display of distribution information provides a good overview of the retrieved set of documents with respect to the free-form user query.

For TREC-4 we were interested in investigating how end-users used the visualization scheme. We were also interested in finding what aspects of the system were frustrating, what aspects they liked and what they wanted in the system. We have yet to do a thorough statistical analysis of the trace data to quantitatively determine the ways in which users with visualization tool acted different from the users without the visualization tool. What we report here is our observations of user interactions, information from structured interviews, and questionnaires.

In the next section we give a brief description of our system. Then we describe our experimental design followed by our observations as it relates to the visualization tool. Then we discuss user's frustrations, likes and wants.

2 System Description

For our study, we used the INQUERY retrieval engine from University of Massachusetts, Amherst [CCH92]. We built a simple graphical user interface on top of INQUERY using Tcl/Tk [Ous94]. There are two versions of our system - one with the visualization, and one without. In our base system, as shown in Figure 1, there are three windows: the top left window is for entering and editing the query. The titles of retrieved documents are displayed immediately below that window. Thirty titles can be displayed in one screen. One can scroll down to a maximum of 150 document titles. Mouse-clicking a title brings up the full text of that document in the window at the bottom right. By clicking the "Next Query Word" button in the full text window, one can position the full text display such that the next occurrence of query word in the document is at the top of the window.

One can save documents and mark documents for relevance feedback by clicking the "Save?" and "Rel?" buttons immediately to the left of the title in the title display window. The only operator that is allowed is the adjacency operator: A hyphen between two words specifies that the two words must appear right next to each other in the same order in a document in order for the word-combination to contribute to the retrieval of that document. There is no negation operator. Automatic stemming and stopping are performed.

The visualization tool is displayed in another window as shown in Figure 2. It

consists of a series of vertical column of bars. There is one column of bars for each document. The leftmost vertical column of bars corresponds to the document ranked 1 and the rightmost vertical column corresponds to the document ranked 150 with all the intermediate ranks lying in between. In each vertical column there are multiple bars - one each for each query word. The height of the bar at the intersection of query word row and a document column corresponds to the weight of that query word in that document. Thus if there are a handful of query words that convey the crux of the query and is very important for a document to contain these query words, one can quickly see from the visualization which retrieved documents have those important words. One can also see how many of the retrieved documents have those words in combination to get a feel for the overall goodness of query results. The effects of modifying the query, like adding a query word, would clearly be shown in the visualization. One can quickly take stock of how useful the query modification turned out. Moving the mouse cursor over the vertical columns would highlight the column directly beneath the mouse cursor and simultaneously highlight the title corresponding to that document in the title display window.

Apart from the query words typed in by the user, the visualization also shows the distribution information for words added by the system due to relevance feedback. In summary, all the words internally used by the system in computing the query results are shown in the visualization. The words in the visualization are also stopped and stemmed.

3 Experimental Setup

The searchers for our study were undergraduate student volunteers from a course on library searching at Georgia Tech. All the searchers had prior computer experience - a majority of them more than 4 years. All the students were majoring in an engineering discipline. They had differing levels of experience with the Georgia Tech Electronic Library catalog - a boolean online public access catalog.

All the users were asked to fill out a background questionnaire. They were given a tutorial on how to use the system. They were then asked to do a practice search on topic 224 for 15 minutes. Following that they were asked to find as many documents as they can that address the given information problem without too much rubbish (as specified by the interactive track guidelines). This was followed by another intermediate tutorial and then a search for a second topic. Immediately after each of the two real searches, they filled out a search evaluation questionnaire. Finally, there was a structured interview.

Figure 1: Sample querying session. The window in the top-left corner is the query entry window. Immediately below that is another window where the titles of retrieved documents are displayed. To the bottom right is another window where the full text of documents are displayed.

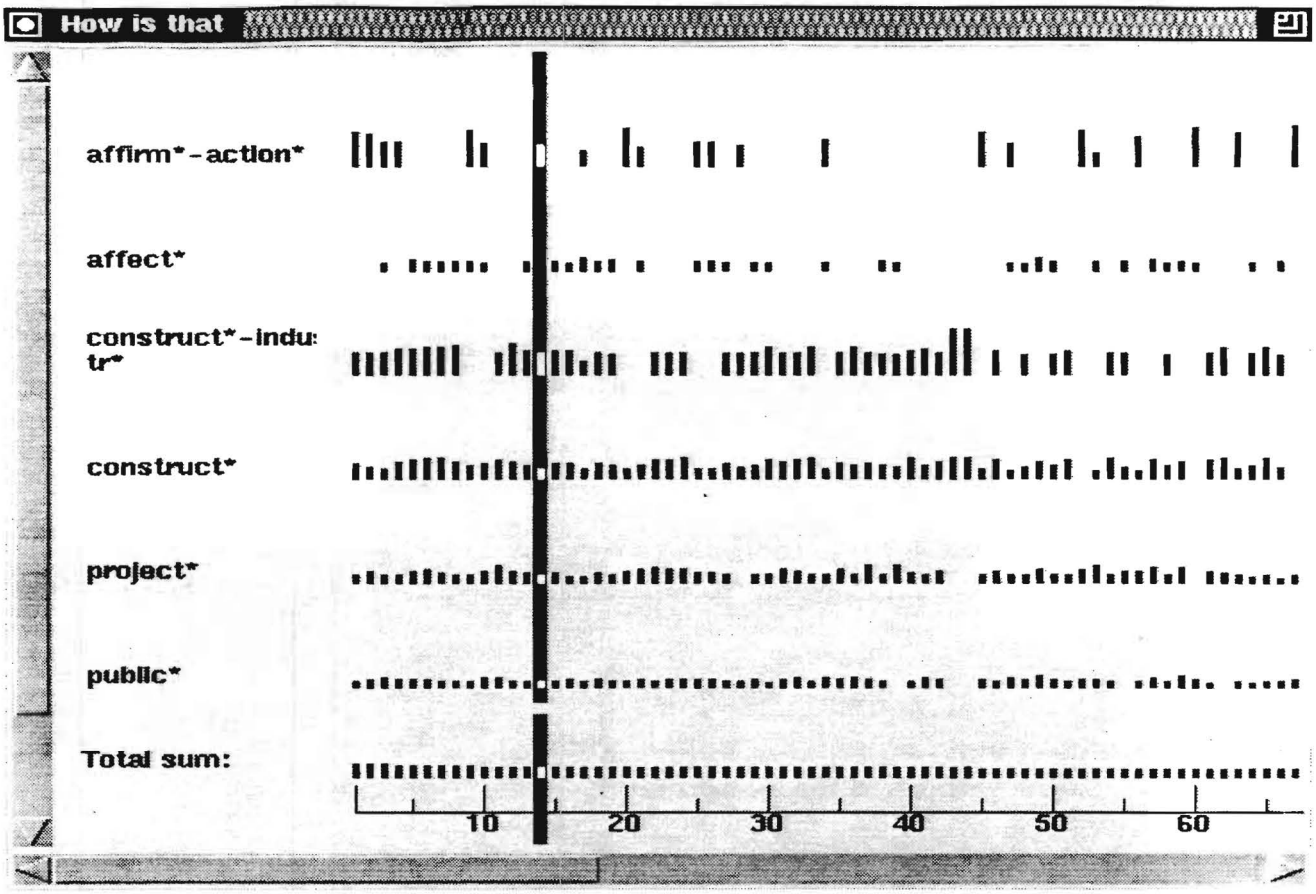


Figure 2: Visualization of results. The highlighted vertical column corresponds to document ranked 14. The title of document ranked 14 document will also be highlighted in the title display window. Clicking the highlighted vertical column brings up the full text of that document.

The searchers were divided into three groups. In each group there were 12 searchers. In the first group (hereafter named "w:w", since both first and search topics are searched WITH visualization), the searchers used the visualization tool for all the searches and the tutorial. In the second group (hereafter named "wo:w", since the first search topic is searched WITHOUT and second topic WITH visualization), the initial tutorial, the practice search and the first search was done without the visualization tool. The intermediate tutorial introduced the visualization tool and the search for the second topic was done with the visualization tool. In the third group (hereafter named "wo:wo", since both the search topics are searched WITHOUT the visualization), all the tutorials and searches were done without the visualization tool. The intermediate tutorial for the w:w and wo:wo groups was a dummy tutorial to compensate for the intermediate tutorial of the wo:w group.

Since each searcher searched for two topics and there were 12 searchers in each group, all the 24 topics were covered by each of the three groups. The 24 topics were randomly divided into 12 pairs and each pair was searched by 3 searchers, one each from the w:w, wo:w and wo:wo groups. The idea was to compare the performance among the three groups to find out the effects of the visualization scheme. Only 24 of the 25 topics for the interactive track were given to end-users in the study. The remaining one topic (topic 223) was searched by the author using the visualization tool.

The searchers were asked to think aloud as they used the system. For the most part, there was an observer in the same room using a different computer and simultaneously observing the searcher. Based on such observations while the user session was in progress, we felt that huge searcher differences in interpreting the query combined with huge differences in the nature of the search topics will greatly confound the effects of the visualization tool. As a result, we decided to run a second study. In the second study, we picked topic 242 for the practice search and the practice search was extended to 30 minutes. The intermediate tutorial was removed. We picked topics 203 and 236 for all the searchers. There were two groups of searchers for the second study – the first group had the visualization tool and second group did not have the visualization tool. There were 18 searchers in each group. By keeping the two search topics constant for all these searchers, we expected to eliminate the effects of search topic difference. It turns out that the searcher variability in interpreting the search topic is so huge among searchers that it is not fair to compare different searchers using different systems unless the search topic is extremely clear and specific.

4 End-users view of the visualization tool

A vast majority of the users mentioned visualization as one of the aspects of the system that they liked. They mentioned using the visualization tools in the following ways.

- Some searchers mentioned using it to see the importance of query words in the retrieved documents - as given by the height of the bar. They mentioned that they were more likely to look at the full text of a document if it has a higher concentration of the important query words.
- Most of the searchers mentioned using it most frequently to see the co-occurrence of important query words in the retrieved documents. They mentioned it being easier to use the visualization tool to look for the co-occurrence information than going through the full text of documents in search of occurrences of the important query words.
- Many searchers felt that the visualization in conjunction with the document title gives a fairly good idea of what the document is about. If the title looks promising and the visualization shows that the document has the right combination of query words, one is tempted to look at the full text of the document.
- They mentioned using it to get a quick overview of the number of retrieved documents a query words appears in. They mentioned using it as a checkpoint to see if a query has turned out the way they had expected it to. If not, they were tempted to readjust the query to get a better result. This happens often when some of the crucial query words are not well represented in the retrieved documents. In that case, one is tempted to add synonyms or words related to those crucial query concepts.
- Some of the searchers mentioned that the visual nature of the distribution information was much easier to identify things than reading text information. This suggests that the mental effort of reading textual information as being much higher than interpreting a simpler visual pattern, and given a choice, the users are more likely to choose the latter.
- **Disadvantages:** A few searchers mentioned that relying heavily on the visualization can also hurt as follows: They mentioned that using the bar-graph to pick out a document containing certain query words may not be indicative of the content of the document - just as the title may not be a good indicator of content. An exemplar case is searcher 35 on the topic of "status of nuclear proliferation treaties". Since almost all of the retrieved documents had something

to do with "nuclear proliferation", the searcher mentioned using the visualization tool to pick those documents containing the query word "status" - only to see that the usage of "status" in the document was not in the context of nuclear proliferation treaties. Then the searcher started paying little emphasis on the presence of "status" in documents. Although relying on that information was initially detrimental, one tends to learn when and how to rely on the visualization. We believe that the presence in retrieved documents of adjectives, adverbs and verbs from the query may not be good content indicators especially when they have a high collection frequency. And relying on the visualization to select documents that have these adjectives, adverbs and verbs from the query may not help.

In summary, the visualization tool seems to help in the following ways:

- to gain more information about specific documents in addition to the title before looking at the full text. Higher concentration of important query words in a document suggests a closer look at the document.
- to gain aggregate information about the query result. The absence of important query words in a vast majority of the retrieved documents suggests query reformulation by adding synonyms and other related concepts.

5 Likes, Frustrations and Wants of users

Apart from the visualization, we were also interested in finding if there are any specific facilities that the users wanted, what features they liked, and what aspects were frustrating. While interpreting the following, we wish to reiterate that all the searchers had some amount of experience with the Georgia Tech Electronic Library catalog which is a character-based-command-driven interface to a boolean system. Some of the features they liked may arise out of the fact that they have had little experience with ranked output systems and the only other major information retrieval system they know is a character based interface to a boolean system.

5.1 Likes

- A vast majority of the searchers with the visualization mentioned that the visualization tool and relevance feedback as the two major aspects of the system they liked. Searchers without the visualization mentioned relevance feedback as the most important feature they liked.

- A number of searchers found the fact that all the information (like the user query, titles of documents and the document full text) is displayed simultaneously in one screen to be very useful. In the Georgia Tech library system, one has to switch between screens to get different types of information. There seems to be a significant mental overload in the context switch between screens. Having simultaneous access to all information seems to bring about a rich interplay between the different sources of information.

- many searchers mentioned that the mouse-based graphical nature of the interface is a significant improvement over a command line based interface.

- many searchers also mentioned that the free-form textual queries without having to worry about any syntax leads to a free flow of thought. "I like the fact that I can type in whatever comes to my mind ... knowing that it will ignore all the junk words like a, an, the, etc...".

- The "Next Query Word" feature was also liked by many searchers. They liked it because they did not have to scroll through a long document looking for occurrences of query words. (All the occurrences of query words in a document are highlighted by the system).

5.2 Frustrations

- A number of searchers mentioned that it was frustrating when the system takes a long time to get the full text of a large document. Similarly, they were also frustrated when it takes a long time to evaluate a query with a large number of relevance feedback documents. The longest delay for evaluating a query was about 2 minutes (when there are about 30 relevant documents). Most of the query evaluations took less than 20 seconds. They said that they understand that the system has to process a lot information (when there a number of relevant documents), but it was frustrating nevertheless.

- Some searchers said that it was frustrating to spend some time reading through the full text of a document and when they are halfway, realizing that they had already seen the same/similar document.

- While some searchers seemed to like having access to 150 retrieved documents, some others mentioned that 150 documents is too much especially when most of the 150 are not relevant. They seem to have the opinion that if some documents are definitely not relevant to the query, then they should not be shown. Thus, this problem is not alleviated even if one reduces the number of documents

displayed. They seem to be quite sensitive about precision. They are not as sensitive about recall – since they are usually satisfied if they get a few documents concerning the topic. Based on our observations, we believe that when the non-relevant documents consistently come from a particular subject area, and when the user is not in a position to remove those documents, they tend to get more frustrated. Using subject classification schemes (where available) to negate disinteresting subject areas would help in this regard.

- In our system, when the title for a document is not available, the message "No title for this document" is displayed instead of the title in the title display window. Many of the federal register documents do not have a title and this is quite annoying to some searchers since they do not have any idea about the document content. This makes it difficult to decide whether to request the full text or not. In cases where the full text is requested, the document happens to be large and hence takes a lot of time to retrieve, thereby adding to the frustration.

- Some federal register documents do not have anything worthwhile – they consist of a listing of subject areas or table of contents. Some searchers wondered why these documents were in the database in the first place.

- Some searchers mentioned a general dislike towards federal register documents partly because they felt that many of them did not have any important piece of information, partly because in general they have no title, partly because it took too long to retrieve them.

- Some searchers were frustrated when a document that they know as non-relevant keeps coming up in the query result. The fact that they were not able to delete the document from the display seemed to add to the frustration.

5.3 Wants

Many of the frustrations mentioned above seemed to directly translate into wants for removing the causes of frustration. In addition to those wants, we observed the following:

- Many searchers expressed a desire to remove certain query words that were added by the system from relevance feedback documents – especially when they are proper names and when they are not necessarily what they are looking for.

- A number of searchers wanted a keyboard equivalent of mouse actions. This is not to say that they did not want mouse actions. It seems to be a significant effort for these searchers to move the right hand out of the keyboard, reach over to the mouse, look at the screen to position the mouse cursor, click the mouse button and move back to the keyboard.

- When asked if they felt a need to have access to an online thesaurus, some searchers expressed a desire for it and some did not. Some of those who did not want a thesaurus mentioned that relevance feedback seemed to alleviate the need for a thesaurus.

- Many searchers wanted to be able to specify that the system should definitely avoid retrieving certain documents in subsequent query iterations. They wanted to have a negative relevance feedback where the system avoids all documents like a particular nonrelevant document.

6 Acknowledgments

The tremendous help from Prof. Nick Belkin regarding the experimental setup and questionnaire design is highly appreciated. Many thanks to Prof. Scott Hudson and Prof. Shamkant Navathe who were instrumental at every stage of the interface development. We appreciate the help of Prof. Jan Crowe who was very cooperative in letting the students of her class participate in the experiments. Special thanks to Prof. Bruce Croft for letting us use the INQUERY retrieval system. Support from the ARPA contract No. F33615-93-1-1338 is appreciated.

References

[CCH92] J.P. Callan, W.B. Croft, and S.M. Harding. The inquery retrieval system. In *Third International Conference on Database and Expert Systems Applications*, September 1992.

[Ous94] John K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, 1994.

Evaluation of a Tool for Visualization of Information Retrieval Results

Aravindan Veerasamy
veerasam@himalaya.cc.gatech.edu
College of Computing
Georgia Tech
Atlanta, GA, USA

Nicholas J. Belkin
belkin@scils.rutgers.edu
School of Communication, Information & Library Studies
Rutgers University
New Brunswick, NJ, USA

Abstract

We report on the design and evaluation of a visualization tool for Information Retrieval (IR) systems that aims to help the end user in the following respects:

- As an indicator of document relevance, the tool graphically provides specific query related information about individual documents
- As a diagnosis tool, it graphically provides aggregate information about the query results that could help in identifying how the different query terms influence the retrieval and ranking of documents.

Two different experiments using TREC-4 data were conducted to evaluate the effectiveness of this tool. Results, while mixed, indicate that visualization of this sort may provide useful support for judging the relevance of documents, in particular by enabling users to make more accurate decisions about which documents to inspect in detail. Problems in evaluation of such tools in interactive environments are discussed.

1 Introduction

The disadvantages of Boolean IR systems are well known. Best-match (i.e. ranked output) systems address several of these problems by allowing users to submit unstructured queries, and by ranking the retrieved documents in (presumed) order of relevance. However, such systems also introduce new problems, or exacerbate problems that are not so severe in Boolean systems.

For instance, understanding why a document (or set of documents) was retrieved is relatively straightforward in exact-match systems, since all members of the set are required to contain exactly the query specification. Furthermore, the ordering within the set of retrieved documents is typically based on relatively well-understood formal characteristics of the documents, such as date of publication or alphabetical order by title or author. In best-match systems, on the other hand, neither the matching rule nor the ranking rule is easily understandable. The former is usually based on characteristics and algorithms which don't have simple relationships with the unstructured query; the latter is intended to reflect complex conceptual relationships between the query and the individual documents, and between the documents themselves.

SIGIR'96, Zurich, Switzerland©1996 ACM 0-89791-792-8/96/08.$3.50

Furthermore, query reformulation may be more difficult in Boolean than in best-match systems. Obtaining a manageable output set size in Boolean systems (the most typical re-formulation task) may be less demanding than attempting to rearrange the list of retrieved documents in a best-match system by manipulating an unstructured query. This is especially difficult when the rules for ordering and matching are not well understood.

Current best-match IR systems take relatively little account of these issues. In response to a user's query, most systems display surrogates (title, source, author ...) of the top 'n' retrieved documents, in a list, with some number(s) indicating the rank, or reason for being in that rank, i.e. a retrieval status value (RSV). Some systems display by default more information about the first retrieved document; most require the user to request such information (e.g. the full text of the document) explicitly. The only explanation of *why* the documents are ranked the way they are is typically the RSV, about which there is no further information than the number itself. More explanation may not be necessary in situations where the top retrieved documents are all clearly relevant. But when the user needs to modify the query in order to get better results, understanding the causal relationship between query and document ranking becomes very important. Having an accurate idea of *why* a list of documents was retrieved, of *how* they were ranked, and of *what* is sub-optimal about the ranking could be useful in effective query reformulation.

Knowledge about the relationships between query and ranking of retrieved documents is not in itself sufficient for effective query reformulation. It is also necessary that the user be able to manipulate the query effectively after the problem has been identified. For instance, knowing that an important query concept is missing in most of the retrieved documents is not sufficient for effective query reformulation. One must then be able to find the right words (or other techniques) for increasing the importance of the concept in the query. Without the ability to take corrective action once the problem is diagnosed, the diagnostic information is of little value.

A possible means for addressing problems of this sort is to display to the user something about the documents which relates them directly to characteristics of the query, and which relates them to one-another. Highlighting query terms in the text display of retrieved documents attempts to accomplish the former, and the indication of RSV is an attempt to accomplish the latter. However, neither of these techniques appears to give sufficient information to guide effective query reformulation. Graphical displays of the characteristics of retrieved documents (*visualizations*) which are relevant to their retrieval and ranking is one obvious approach to this problem.

A further problem in IR systems in general has to do with the multi-stage nature of presentation of results. The initially-presented surrogates are meant to provide a concise picture of what a document is about. Based on these surrogates, the user

may request more detailed information about particular documents which look promising (or for which the surrogate information is equivocal). In some cases, this might be the "full" bibliographic information about the item, in others an abstract, and in many systems now, the full text of the item. Thus, as the user progresses through the stages of display, that which is displayed is more complete and informative, allowing increasingly accurate relevance judgments. But, the information in the later stages of display is also more time-consuming to peruse. Therefore, it is important for the searcher to be reasonably certain that it is worthwhile doing this inspection. The information displayed in the earlier stages thus serves as a filter which supports the user in deciding which documents do not need further inspection (either because they are obviously good or obviously bad), and which documents do justify the further effort.

Thus, displaying a great deal of information at the surrogate stage of display could be a useful device for judging the relevance or usefulness of the document. The advantage is that when the user requests the second-stage display, it is more likely that that document will be relevant to the user than if there were less information in the first stage. The disadvantages to this strategy are that since there is more text to display in the first stage, fewer items can be presented, and more time must be spent in perusing the first-stage display. Thus, the total number of documents seen by the user may well be fewer, although the quality of the decision-making may be higher.

If, on the other hand, one chooses to display less information at the initial surrogate stage, then the decision about whether to look at the more complete display is less secure. Hence, the proportion of second-stage documents which turn out to be relevant is likely to be low. The advantage of seeing more documents, more quickly, in the first stage is thus offset by the additional time that is spent perusing non-relevant documents in the second stage.

A possible means to addressing this problem is to display information about the document in the first stage in some form that does not require as much perusal time and screen space as text. Graphical displays of the characteristics of documents which are significant in supporting the decision to peruse or not (*visualizations*), could support set-at-a-time perusal of documents, rather than document-at-a-time perusal of text displays.

It will not escape the reader that the suggested solutions to the two classes of problems that we have raised here are rather similar, and could, indeed, be instantiated by the same sort of display. We present a visualization tool which is intended to address these problems in IR systems, and a preliminary evaluation of this tool. The remainder of this paper is organized as follows. We first present a description of the visualization tool, and a rationale for the features of this tool with respect to the problems in IR interaction that we have discussed. We then discuss some related work in IR visualization that addresses this type of problem, and draw some comparisons between that work and ours. We follow with a description of the experiments we conducted to evaluate the visualization tool, and the results of those experiments. We conclude with some comments on the implications of our results, on future work, and on the implications of our evaluation experience for evaluation of interactive IR in general.

2 Visualization tool

2.1 Description

The visualization tool is an adjunct to a basic interface for IR. This interface is structured as a indicated in Figure 1, with a query window, a display of titles retrieved, and the full text of a document. This serves as the baseline interface interaction which is compared to the visualization tool. A screen snapshot of the visualization tool is shown in Figure 2. The visualization corresponds to the query "how has affirmative-action affected the construction-industry construction projects and public works".

The visualization consists of a series of vertical columns of bars. There is one column of bars for each document. The left-most vertical column of bars corresponds to the document ranked 1 and the rightmost vertical column corresponds to the document ranked 150. In each vertical column there are multiple bars -- one each for each query word. The height of the bar at the intersection of query word row and a document column corresponds to the weight of that query word in that document. Moving the mouse cursor over the vertical columns highlights the column directly beneath the mouse cursor and simultaneously highlights the title corresponding to that document in a title display window.

Apart from the query words typed in by the user, the visualization also shows the distribution information for words added by the system due to relevance feedback. Thus, all the words internally used by the system in computing the query results are shown in the visualization. This window is scrollable, in case the number of words in the query exceeds the vertical space. The words in the visualization are also stopped and stemmed. The basic interface, and the visualization tool, utilize the INQUERY retrieval engine, version 2.1p3 [Callan, Croft, Harding, 1992]. We use all of the default features of that system, including their relevance feedback, stemming and stoplist algorithms, but do not use any of the structured query facilities.

2.2 Response to problems of IR interaction

In support of query reformulation, the visualization makes the connection between the query and retrieved documents explicit by graphically displaying the contribution of each query word to the retrieval of each document. The higher the contribution of a particular query word to the retrieval of a document, the taller the bar at the intersection of the corresponding query word and document. The absence of a bar at the intersection illustrates the absence of the term in the document. Absence of an important query concept in a number of retrieved documents points to a problem situation which the user needs to work on. The visualization makes relations between the documents themselves explicit, since the characteristics which have led to their rank (the number and contribution of matching terms) are explicitly displayed.

In support of informative first-stage display, the visualization provides a great deal of information useful for deciding whether to view the full text of a document in a highly condensed way, and allows many document surrogates to be displayed at one time. The presence or absence of specific significant words in any document can be quickly seen, and it is possible to identify sequences of documents which do, or do not have important contributions from specific query words.

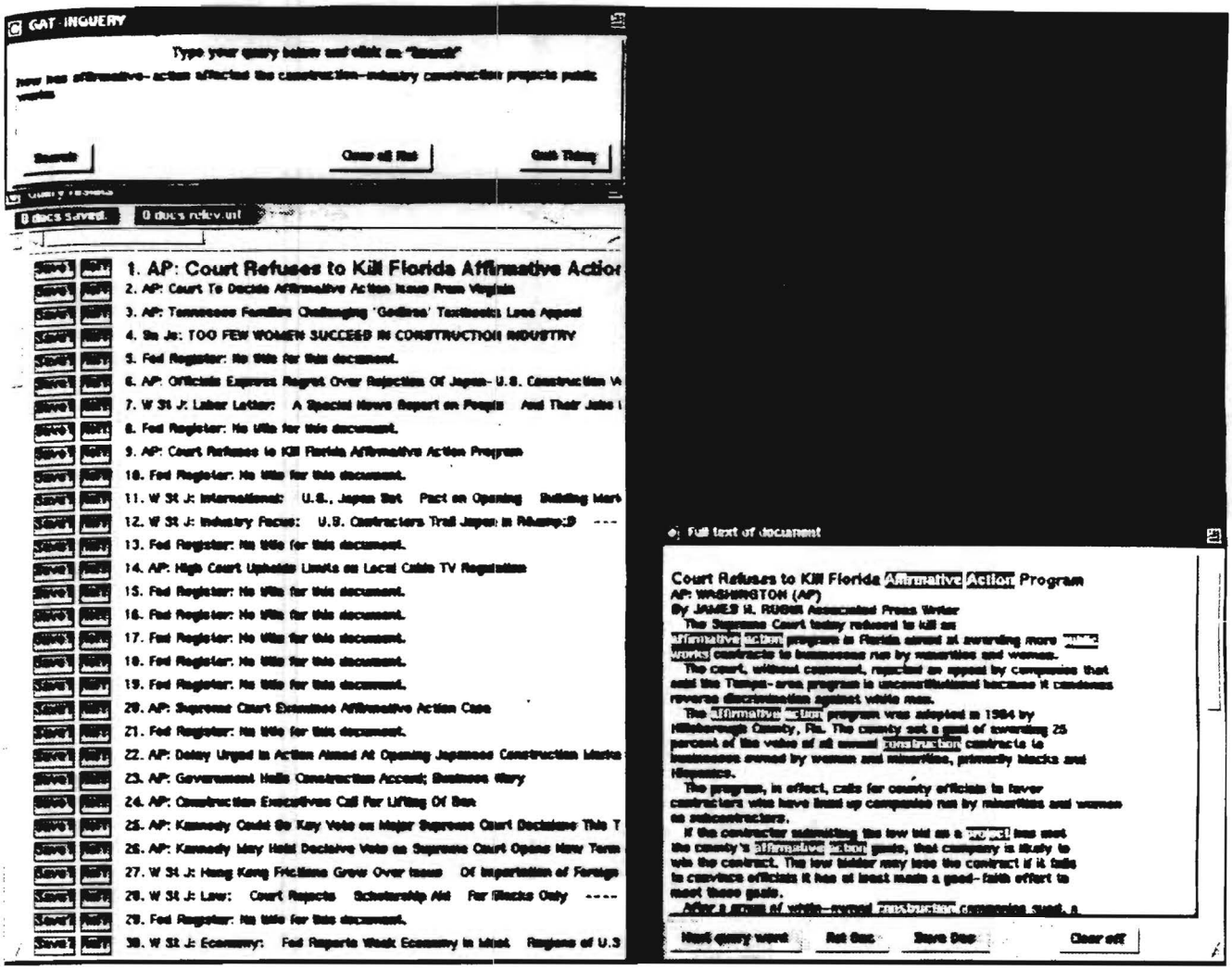


Figure 1. Sample querying session. The window in the top-left corner is the query entry window. Immediately below that is another window where the titles of retrieved documents are displayed. To the bottom right is another window where the full texts of documents are displayed.

For the example search topic ("How has affirmative action affected the construction industry?"), there are two facets that are central: "affirmative action" and "construction industry". From the visualization tool, we can immediately see that most of the documents are concerned with the "construction industry" and only a portion of the documents have the term "affirmative action". We can also see that the "affirmative action" concept is spread sparsely throughout the top 70 documents. The graphical format of presentation has some important advantages in that it is more condensed than an equivalent text display.

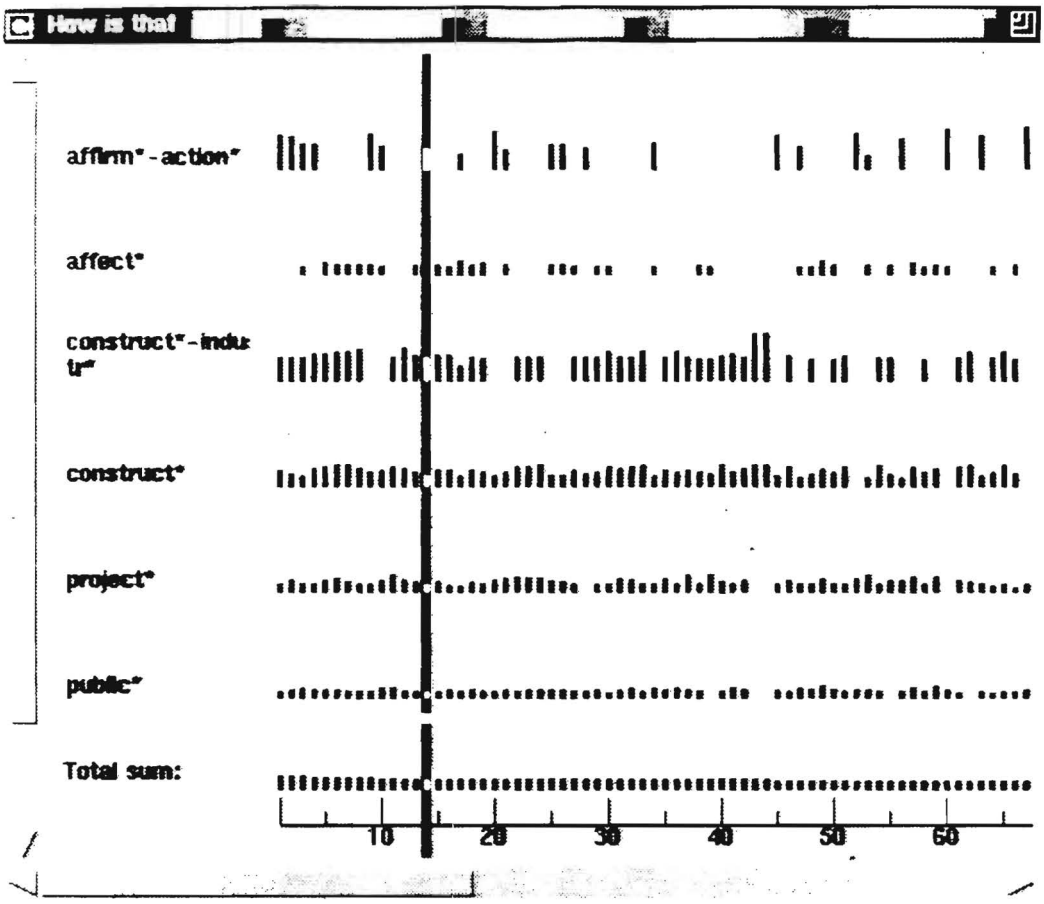


Figure 2. Visualization of results. The highlighted vertical column corresponds to document ranked 14. The title of document ranked 14 document will also be highlighted in the title display window. Clicking the highlighted vertical column brings up the full text of that document

From the visualization, one gets an immediate idea of how the different query words influence the document ranking (as given by the height of the bars). One can see that the concept "affirmative action" is not well represented in the retrieved documents. This suggests that synonyms and words related to that concept must be added to the query to reinforce that query concept in subsequent search iterations. From the visualization tool, one can infer that the system interprets "public" and "project" as two separate words and that the contribution of those two words to the retrieval of almost all documents is uniformly low (as given by the height of the bars). One can probably improve the situation by making "public projects" a phrase, thereby retrieving documents that have these two words in close proximity. Gaining such overall information about the query results by reading the document text is at best cumbersome if at all possible.

3 Related visualization work

A number of visualization schemes for information retrieval systems have been proposed. The Perspective Wall [Card, Robertson & Mackinlay, 1991] is a visualization scheme which supports browsing of documents. While such a system can not handle qualitative document classifications such as library subject catalogs, it is very useful for visualizing documents based on data which are linear in nature (like date of publication). A nice way of integrating different visualization schemes for efficient navigation through the hypermedia space has been proposed by [Mukherjea (1995)]. These schemes are primarily useful for navigational tasks. Other visualization schemes such as those of [Korfhage (1991)], [Spoerri (1994)], [Hemmje, Kunkel & Willett (1994)] have facilities for viewing a large document space. But visualizing the document space along more than 3 - 4 dimensions simultaneously becomes

very cumbersome using their systems. The visualization scheme in our tool can gracefully handle more query word dimensions. Many systems are tailored towards easy construction of queries [Spoerri (1994)] [Aboud, et al. (1994)] [Arents & Bogaerts (1993)] but do not pay much attention to the display of query results.

TileBars [Hearst (1995)] visually shows the query term distribution and overlap in retrieved documents. The term distribution in retrieved documents is shown right beside the title of the document. In a number of respects, the reasons and motivations for Hearst's work are similar to those of our visualization [Veerasamy, Navathe & Hudson (1995)] [Veerasamy & Navathe (1995)] [Veerasamy (1996)]. There are some important ways in which TileBars differs from the visualization that we propose.

- TileBars provides information on how different query facets overlap in different sections of a long document. Our visualization scheme does not provide information at that fine level of granularity.
- TileBars presents the document surrogates in a list, making it more difficult than in our tool to gain an overall picture of the query word distribution for a whole set of documents in one glance.
- TileBars seems best suited for long documents, while our visualization scheme does not seem to be constrained by length.

4 Experiments to test the effectiveness of visualization

4.1 General conditions

We discuss two experiments for testing the effectiveness, usability and acceptability of the visualization tool by comparing searching with an interface using the visualization, versus searching with the same interface, but without the visualization tool. The underlying retrieval engine used in these experiments was INQUERY version 2.1p3, from the University of Massachusetts, Amherst, generously made available to us by Prof. Bruce Croft [Callan, Croft & Harding (1992)]. We developed the graphical user interface using Tcl/Tk on top of INQUERY.

The experiments were conducted as part of the TREC-4 interactive track (Harman, 1996). Thus, the task for the searchers in the experiment was the TREC-4 interactive track task:

> Find as many documents as you can which address the given information problem, but without too much rubbish. You should complete the task in about 30 minutes or less.

The "information problems" were chosen from the 25 adhoc topics used for the TREC-4 interactive track, and the database was the TREC Disks 1 and 2 database of the full texts of about 550,000 documents.

Both experiments were designed to test the usefulness of the visualization tool for addressing the two problems that we have discussed and that motivated the design of the tool:

- efficiency and effectiveness in discovering relevant documents; and,
- effectiveness in supporting query reformulation.

In order to test the former, we predict that searchers using the visualization tool will make better decisions about which documents to look at (or not look at) than those without visualization. We operationalize this difference with the following dependent variables:

- the number of documents saved per search (*s-p-s*). Since search times are more-or-less constant (about 30 minutes) across searchers, this measure reflects efficiency in being able to see more documents.
- the proportion of documents whose full text was viewed that were judged relevant by TREC evaluators (interactive trec precision or *i-t-p*). This measure indicates the quality of the documents which were chosen for viewing.
- the proportion of documents whose full text was viewed that were saved by the searcher (interactive user precision or *i-u-p*). This measure also indicates quality of documents which were chosen for viewing, but is indicative of the relationship of the display to the searcher's own concept of relevance to the problem, rather than being dependent upon the external relevance judgments.

To test the latter, we use:

- precision of the search, measured in the required manner for the TREC-4 interactive track; that is, as the proportion of documents saved by the searcher that were judged relevant by the external judges. This measure is indicative of the effectiveness of retrieval performance.

For all of these measures, higher numbers mean better performance.

The subjects for both experiments were undergraduate student volunteers who were registered in a one-credit hour course on library searching in the College of Computer Science at Georgia Tech. All subjects had prior computer experience, the majority with more than four years. All subjects were majoring in an engineering discipline, and had varying levels of experience with the Georgia Tech Electronic Library Catalog. They had no other IR experience than that offered by the class. Two different groups of subjects were used in the two different experiments.

All the subjects in both experiments followed the same general introductory and tutorial procedure. They were asked to fill out a background questionnaire about their computer and IR experience, major, and so on. They then had a hands-on tutorial (about 1 hour) on how to use the version of the system they would be using for the first experimental search. They were then asked to do a practice search on TREC topic 224 ("What can be done to lower blood pressure for people diagnosed with high blood pressure? Include benefits and side effects.") for 15 minutes. They then did the assigned searching tasks (details differ between the two experiments), during which they were instructed to "think aloud", which was recorded on audio tape. All the user interaction with the system was logged. After each search, they completed a search evaluation questionnaire. At the end of the session, a structured interview on their use of the system was administered. All subjects did three runs of the system: one practice run and two test runs.

4.2 Experiment 1

Thirty-six subjects were randomly divided into three groups of twelve each. Twenty-four of the 25 TREC-4 interactive track topics were randomly divided into twelve pairs. Each of the twelve pairs of search topics was randomly assigned to one of the searchers in each group, one to be searched in the "first" condition, the other to be searched in the "second" condition for the group of which the searcher was a member. The topic pairs were searched in the same order in all groups. Thus, the same twelve of the 24 topics were searched in the first condi-

tion for all three groups, and the other twelve were searched in the second condition for all of them.

The three groups were defined according to the combination of conditions or treatments. Group wo:w (for WithOut:With) did the initial tutorial, the practice search and the first search without the visualization tool. An intermediate tutorial after the first search introduced the visualization tool and the search for the second topic was done with the visualization tool. Group "w:w" (for With:With) used the visualization tool for all the searches and the introductory tutorial. Group "wo:wo" (for WithOut:WithOut) did all the searches and the introductory tutorial without the visualization tool. In both the w:w and wo:wo groups, an intermediate tutorial on the interface with which they were working was introduced between the two searches to match the intermediate tutorial of the wo:w group.

This "within subjects" design was used in order to control for user differences, and to account for any possible learning effects from search 1 to search 2. It was predicted that performance on the various measures would improve from first search to second search in the wo:w group, more than in either the wo:wo or w:w groups.

4.3 Experiment 2

In this experiment, 36 subjects were randomly divided into two groups, one with the visualization tool ("viz"), the other without ("noviz"). Three search topics were chosen for searching by all eighteen searchers in each of the two groups, always in the same order. The searchers in the two different groups followed the same pattern of participation as those in experiment 1, but without any intermediate tutorial, and with the practice search time extended to 30 minutes. We picked topic 242 ("How has affirmative action affected the construction industry?") for the practice search. The first "experimental" search was on topic 236 ("Are current laws of the sea uniform? If not, what are some of the areas of disagreement?"), and the second was topic 203 ("What is the economic impact of recycling tires?").

This "between-subjects" design was used to control for the effects of search topic difference, and to have larger numbers of subjects in the two conditions. It was predicted that performance in the viz group would be better than performance in the noviz group for each topic.

5 Results

In this paper, we report only on results with respect to the performance measures we have defined. Results from the questionnaires with respect to use and usability of the two systems, and with respect to interaction measures and "thinking aloud" will be reported in subsequent publications.

5.1 Experiment 1

The results of experiment 1, displayed in Table 1, are something of a disappointment. There are no significant differences (using the Wilcoxon Matched-Pairs Signed-Ranks Test, one-tailed at $p \leq .05$) between any of our four measures between the without- and with-visualization treatments in the wo:w group. Furthermore, there are no significant differences between any of the matched without-visualization/visualization groups (i.e. between the second searches of wo:w and wo:wo groups, the first searches of wo:w and w:w groups, and both searches of the wo:wo and w:w groups). There is no consistent pattern on any of these measures from first to second search (i.e. there appears to be no learning effect, nor does it appear that one of the sets of twelve topics is in general more difficult than the other, nor is it the case that any of the groups does consistently better or worse for either search). These very mixed results lead us to think that our experimental design in this case suffers from two significant problems. The first is great inter-subject and inter-topic variability; the second is that we have too few subjects for each condition to adequately test significance of any differences that may exist.

Condition	Precision			s-p-s			i-t-p			i-u-p		
	Mean	Median	SD	Mean	Median	SD	Mean	Median	SD	Mean	Median	SD
wo:w.1	0.712	0.750	0.257	2.193	1.630	2.671	0.491	0.505	0.296	0.459	0.435	0.210
wo:w.2	0.531	0.635	0.402	2.686	1.915	2.848	0.419	0.405	0.270	0.454	0.445	0.204
wo:wo.1	0.566	0.530	0.286	4.039	1.250	7.289	0.385	0.415	0.187	0.429	0.420	0.163
wo:wo.2	0.513	0.670	0.342	3.500	1.500	5.437	0.344	0.315	0.302	0.379	0.425	0.201
w:w.1	0.586	0.515	0.284	2.319	2.170	1.301	0.402	0.285	0.251	0.443	0.450	0.164
w:w.2	0.496	0.415	0.356	2.480	1.560	3.096	0.433	0.385	0.299	0.407	0.390	0.241

Table 1. Summary results for experiment 1. w = with visualization, wo = without visualization. The order of w and wo in the condition column indicates the order of application of conditions for that group, the number following the two indicates for which of the two conditions, first or second, the value is given. s-p-s = documents saved per search; i-t-p = interactive trec precision; i-u-p = interactive user precision.

5.2 Experiment 2

The results of experiment 1 led to the design of experiment 2, whose results are displayed in Table 2. The rows in Table 2 are in the order that the topics were searched. In order to test the significance of these results, it is necessary to compare them topic-by-topic, without cumulation, to maintain the assumption of independence, since each searcher did three searches (including the practice search, 242) in the same condition. To test for significance of results, we used the Mann-Whitney U test with $p \leq .05$, one-tailed. For precision,

there is no significant difference between nonviz and viz for any of the three topics. For s-p-s, the trend is in favour of viz in all three cases, but significantly so at the chosen level only for topic 242 (although for topic 236 it only just misses). For i-t-p, again the trend is nominally in favor of viz, but is again significant only for topic 242. For i-u-p, the same trend holds, and again viz is significantly better than noviz only for topic 242.

For three of the four measures we can see that there are obvious topic differences which cannot be accounted for by a learning effect, since the direction is wrong. Two points are important to note here. First, it appears that topic 242 was "easier" than the other two topics, and that topic 242 benefited most from visualization. Second, it is clear that differences in topics are likely to affect results averaged over topics, unless there are also quite large numbers of searchers for each topic.

Condition/ Topic	Precision			s-p-s			i-t-p			i-u-p		
	Mean	Median	SD	Mean	Median	SD	Mean	Median	SD	Mean	Median	SD
noviz 242	0.912	1.000	0.140	1.084	1.000	0.706	0.371	0.365	0.132	0.344	0.325	0.132
viz 242	0.898	0.880	0.113	1.899	1.750	0.888	0.545	0.530	0.143	0.454	0.410	0.177
noviz 236	0.227	0.145	0.232	1.347	1.210	0.939	0.085	0.045	0.101	0.376	0.390	0.145
viz 236	0.215	0.140	0.244	2.249	1.670	1.701	0.105	0.080	0.080	0.472	0.440	0.174
noviz 203	0.424	0.470	0.178	1.326	1.085	0.956	0.217	0.215	0.096	0.347	0.330	0.161
viz 203	0.392	0.330	0.226	1.676	1.500	1.205	0.259	0.280	0.070	0.357	0.370	0.157

Table 2. Summary results for experiment 2. viz = with visualization, noviz = without visualization. The numbers following the condition designation indicate the topic searched. s-p-s = documents saved per search; i-t-p = interactive trec precision. i-u-p = interactive user precision.

Interpreting these results is somewhat difficult, although they are a bit more promising than those of experiment 1. It is of some interest that only topic 242 showed significant differences between the noviz and viz groups. This might be explained by that topic's being for some reason more suited to visualization than the other two. Although the numbers of relevant documents for the three queries are rather similar (242: 38; 236: 43; 203: 33), on the basis of median precision reported by all of the TREC-4 interactive track participants, topic 242 is "easier" than topics 203 and 236 (0.2368 vs 0.1515 vs 0.0465, respectively). This of course follows the pattern of precision results by the searchers in experiment 2, but it is not clear how this would explain the apparently beneficial effect of visualization for this topic. An alternative explanation might be that visualization of this sort is helpful for naive searchers, but loses its effect as they become more experienced with the IR system. On the basis of the data we have available, there is no way to decide between these alternatives.

In any event, it seems reasonable to accept, on the basis of the results of experiment 2, that there could indeed be some value to visualization of the sort we have tested here. However, this statement certainly must be very tentative, and subject to much more testing. The results of experiment 1 do not lead to any such conclusion. It must be said, however, that the very mixed nature of these results may well be an effect of the experimental design, and in particular of the inability to take proper account of what may be very large topic differences and searcher differences. Of course, another possible reason for the seeming lack of effect of visualization is the implementation that we chose. This issue needs further investigation.

6 Conclusions

The study reported here intended to demonstrate the potential of visualization to support particular kinds of interactions in IR, and to test one implementation of such visualization. Although the results of our experiments are mixed, it appears that some of them are positive enough to justify further such experiments. But there are some other serious implications of our results.

We are not aware of other work reporting comparisons of visualization tools for IR with equivalent non-visualization interfaces. Our experience suggests that it is important to conduct more such studies, in particular to move beyond assuming the efficacy of visualization to demonstrating it in experimental environments. Our study also demonstrates the severe problems that arise in conducting interactive IR experiments. These include the problems of finding enough subjects to account for inter-subject differences, and of being able to account for inter-topic differences. Balancing these two demands is an exceedingly difficult problem, which is currently severely exercising the TREC-5 interactive track participants.

Another evaluation problem raised by our study is how to measure the effectiveness of visualization tools. The problems with using precision as a measure for evaluating interactive IR are now well-known, especially if precision is decided according to relevance judgments from experts, rather than the searchers. It is also the case that for certain functions of visualization, precision is an inappropriate measure. But we do not have available a suite of accepted alternative measures for evaluating the effectiveness of systems with respect to these functions. So it was necessary for us to invent some new measures which appear appropriate to the IR tasks that we wished to support. Whether these were good choices also needs to be further investigated.

In conclusion, we find that this study has given some support for the general idea of visualization as a tool for enhancing user interaction with search results, and for the specific tool with which we implemented this idea. We also find that the level of support for these statements from this study is not high, and that it is necessary to conduct further studies, with better designs, before we can become confident in the value of visualization for these purposes, as opposed to other tools for interaction. Finally, we find that our study has shown, again, the necessity of developing better measures and methods for the evaluation of interactive IR systems, and the

necessity of rigorous comparative evaluation of visualization in IR.

Acknowledgments

Support from the ARPA contract No. F33615-93-1-1338 to the first author is appreciated. The work of the second author was in part supported by NIST Cooperative Agreement No. 70NANB5H0050.

References

[Aboud, et al., 1993] Aboud, M., Chrisment, C., Razouk, R. and Sedes, F. (1993) Querying a hypertext information retrieval system by the use of classification. *Information Processing Management*, v. 29, 387-396.

[Arents & Bogaerts, 1993] Arents, H.C. and Bogaerts, W.F.L. (1993) Concept-based retrieval of hypermedia information -- from term indexing to semantic hyperindexing.} *Information Processing Management*, v. 29, 387-396.

[Callan, et al., 1992] Callan, J., Croft, W.B. and Harding, S. (1992) The INQUERY retrieval system. In: *DEXA-3: Third International Conference on Database and Expert Systems Applications*

[Card, et al., 1991] Card, S., Robertson, G. and Mackinlay, J. (1991) The information visualizer, an information workspace}. In: *CHI '91: Proceedings of CHI 91 Human Factors in Computer Systems.*. New York: ACM.

[Harman, 1996] Harman, D. (1996) *TREC-4, Proceedings of the fourth Text REtrieval Conference*. Washington, DC: GPO.

[Hearst, 1995] Hearst, M.A. (1995) TileBars: Visualization of Term Distribution Information in Full Text Information Access} In: *Proceedings of CHI 95*. New York, ACM.

[Hemmje, et al., 1994] Hemmje, M., Kunkel, C. & Willett, A. (1994) LyberWorld -- A visualization user interface supporting full text retrieval, In: *SIGIR '94, Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*. London: Springer Verlag, 249-259.

[Korfhage, 1991] Korfhage, R (1991) To see, or not to see -- Is that the query? In: *SIGIR '91: Proceedings of the 14th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*. New York: ACM, 134-141.

[Mukherjea, et al., 1995] Mukherjea, S., Foley, J. and Hudson, S. (1995) Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views. *In CHI 95. Proceedings of the Conference on Human Factors in Computing Systems*. New York: ACM.

[Spoerri, 1994] Spoerri, A. (1994) InfoCrystal: A visual tool for information retrieval and management.. In: *CHI '94: Human Factors in Computing Systems Conference Companion*. New York: ACM, 11-12.

[Veerasamy, 1996] Veerasamy, A. (1996) Interactive TREC-4 at Georgia Tech. In: Harman, D., ed. *The Fourth Text REtrieval Conference (TREC-4)*. Washington, DC: GPO, in press.

[Veerasamy & Navathe, 1995] Veerasamy, A. & Navathe, S. (1995) Querying, Navigating and Visualizing a Digital Library Catalog. In: *Proceedings of the Second International Conference on the Theory and Practice of Digital Libraries*

[Veerasamy, et al., 1995] Veerasamy, A., Navathe, S. and Hudson, S. (1995). Visual Interface for Textual Information Retrieval Systems. In: *Proceedings of the Third Conference on Visual Database Systems. IFIP 2.6*

Effectiveness of a graphical display of retrieval results

Aravindan Veerasamy[1]
veerasam@cc.gatech.edu
College of Computing
801, Atlantic Drive
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
Phone: 404-894-8791
Fax: 404-894-9442

Russell Heikes
russell.heikes@isye.gatech.edu
Statistics Center
School of Industrial Systems and Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

We wish to have this paper considered for the Best Student Paper Award. Veerasamy is a full-time PhD student at Georgia Tech.

[1]To whom correspondence about this paper should be addressed

Effectiveness of a graphical display of retrieval results

Abstract

We present the design of a visualization tool that graphically displays the strength of query concepts in the retrieved documents. Graphically displaying document surrogate information enables set-at-a-time perusal of documents, rather than document-at-a-time perusal of textual displays. By providing additional relevance information about the retrieved documents, the tool aids the user in accurately identifying relevant documents. Results of an experiment evaluating the tool shows that when users have the tool they are able to identify relevant documents in a shorter period of time than without the tool, and with increased accuracy. We have evidence to believe that appropriately designed graphical displays can enable users to better interact with the system.

1 Introduction

The overall concern of all components of an IR system is to present the user as much relevant information as possible. While there has been a lot of work on effective algorithms for retrieving and ranking relevant documents, not much attention has been paid to study the effectiveness of user interface components of IR systems. Apart from retrieval mechanisms, interactive IR systems must also be concerned with the design of appropriate display mechanisms that present the retrieved information in the "best possible manner". We discuss what constitutes "best possible" display by examining a typical user interaction with an IR system. A typical interaction with

current IR systems proceeds as follows:

- User in an Anomalous State of Knowledge [BOB82] expresses his information need as a query that is interpretable by the system.
- The system matches the query with the stored documents and retrieves a set of documents. In the case of ranked output systems, the result is ranked in the decreasing order of relevance. Boolean systems may rank the documents in a chronological order.
- At the *first stage* of display, a set of document surrogates for the retrieved documents are displayed to the user. These surrogates typically consist of a combination of titles, author, source, date of publication, etc.
- The user inspects the document surrogates and requests more information (such as the full text if available) about those that look relevant. This leads to a *second stage* of display that provides as much information about the document (in many cases, the complete document itself) as is available in the system.
- After going through a sufficient number of documents, the user quits the session or reformulates the query to retrieve a better set of documents.

In this scheme, the first stage display of document surrogates is meant to provide a concise and accurate indication of document content. The second stage display of documents provides more information about the document. In cases where the document full text may not be available for the second stage (such as a typical online library catalog), users proceed to a third stage where they examine a paper-copy in library bookshelves where the complete document may be available.

Thus as the user progresses from the initial to the later stages of display, that which is displayed is more complete and informative, allowing increasingly accurate relevance judgments. However, since more information is displayed about a document in later stages of display, they are also more time-consuming to peruse. Furthermore, requesting second stage of display may be more costly since some systems charge a certain fee to deliver the full text of documents. Apart from human frustration of waiting for the delivery of full text, one may have to pay for it since certain systems charge the user based on connect-time. Therefore, it is important for the searcher to be reasonably certain that it is worthwhile requesting this second stage of display.

For the user to make accurate relevance judgments based on the first stage display, the form and content of first stage of display should provide good indication of what document is about. The nature of the first stage display should be such that it can be perused in the shortest possible time – the purpose of the first stage display (of providing a quick and concise indication of document content) is lost otherwise.

Displaying a great deal of information at the first stage of display could be a useful device for judging the relevance or usefulness of the document. The advantage in displaying more document information in the first stage is that when the user requests the second stage display, it is more likely that the document will be relevant to the user. The disadvantage is that since there is more information (and hence more text) to display in the first stage, only fewer items can be presented due to limitations of screen real-estate, and more time must be spent perusing the first stage display. Thus, the total number of documents seen by the user may well be fewer, although the quality of decision-making may be higher.

If, on the other hand, one chooses to display less information in the first stage, then the decision about requesting second stage display is less secure. Hence, the proportion of second-stage documents which turn out to be relevant is likely to be low. The advantage of seeing more documents, more quickly, in the first stage is thus offset by the additional time that is spent perusing non-relevant documents in the second stage.

A possible means to addressing this problem of displaying more information in the first stage without increasing perusal effort and perusal time is to display information in some form that does not require as much perusal time and screen space as text. Graphical displays (*visualizations*) of the characteristics of documents which are significant in supporting the decision to peruse or not, could enable set-at-a-time perusal of documents, rather than document-at-a-time perusal of text displays.

In the remainder of this paper, we describe a visualization tool meant to address this issue; describe and present the results of an experiment evaluating the tool; and draw some conclusions about its effectiveness as a first stage display.

2 Visualization tool

The visualization tool is an add-on to a basic interface for an IR system. There is a query window. The titles and ranks of retrieved documents (first stage of display) is shown below the query window. Figure 1 shows the visualization tool corresponding to the query "How has affirmative-action affected the construction-industry, construction projects and public works".

The visualization consists of a series of vertical columns of bars. There is one

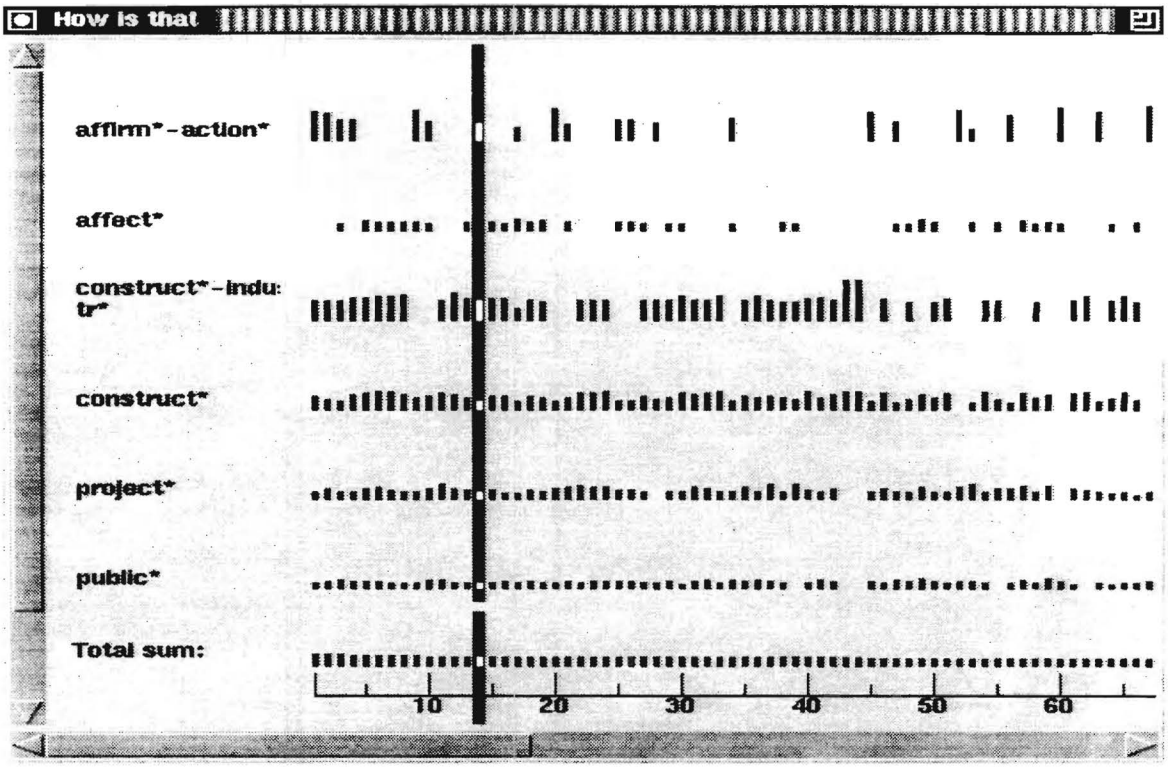


Figure 1: Visualization of results. The highlighted vertical column corresponds to document ranked 14. The title of document ranked 14 document will also be highlighted in the title display window. Clicking the highlighted vertical column brings up the full text of that document.

column of bars for each document. The left-most vertical column corresponds to the document ranked 1 and the right-most vertical column corresponds to the document ranked 150. In each vertical column there are multiple bars – one each for each query word. The height of the bar at the intersection of a query-word-row and a document-column corresponds to the weight of that query word in that document. Moving the mouse cursor over the vertical columns highlights the column directly beneath the mouse cursor and simultaneously highlights the title corresponding to that document in the title-display window. The visualization window is scrollable, in case the number of query words exceeds the available vertical space. The words in the visualization are also stopped and stemmed. Thus the combination of the visualization tool and the title display forms the first stage of display in our system. The basic interface, and the visualization tool utilize the INQUERY retrieval engine, version 2.1p3 [CCH92].

2.1 Response to the need for a concise display of document content

In the Introduction, we discussed the need for a concise first stage display which can also be perused quickly. We believe this visualization scheme to qualify for such a first stage display. It provides information valuable in deciding the relevance of document such as the weight of query concepts in the retrieved documents. The information is also displayed in a highly condensed way, and allows many document surrogates to be perused at one time. While textual display of document surrogates force the user to peruse them a document-at-a-time, this visualization lets the user peruse document surrogates a set-at-a-time. The presence or absence of specific significant words can be quickly seen, and it is possible, in one glance, to identify sequences of documents which

do, or do not have important contributions from specific query words. For the example search topic ("How has affirmative action affected the construction industry?"), there are two facets that are central: "affirmative action" and "construction industry". From the visualization tool, we can immediately see that most of the documents are concerned with the "construction industry" and only a portion of them have the term "affirmative action". We can also see that the "affirmative action" concept is spread sparsely throughout the top 70 documents. The graphical format of presentation has some important advantages in that it is more condensed and can be more easily and quickly perused than an equivalent text display.

3 Related work

A number of visualization schemes for information retrieval have been proposed [CRM91, MFH95, Kor91, Spo94, HKW94, ACRS93, AB93] But most of these do not address either the display of query results or the problem of support of relevance assessment. An exception is TileBars [Hea95], but there are some important ways in which TileBars differs from the visualization proposed here.

- TileBars provide information on how the different query facets overlap in different sections of a long document. Our visualization scheme does not provide information at that fine levels of granularity.

- To make the best use of such additional information in TileBars, the user has to decompose the information need into more-or-less orthogonal facets of a query. However, in our visualization, the user can type in the information need as a free-form textual query.

- TileBars presents the document surrogates in a list, making it more difficult than in our tool to gain an overall picture of the query word distribution for a whole set of documents in one glance.

- TileBars seems best suited for long documents, while our visualization scheme does not seem to be constrained by length.

There are a handful of studies that have investigated the effectiveness of document surrogates as content-indicators to enable human relevance judgments [Jan91, Sar69, RRS61, Tho73, MKB78]. None of them studied the effectiveness of graphical displays (visualizations) of document surrogates as content indicators. A result common to all of these studies is that "accuracy" in relevance judgments increases with increasing information (e.g. Title < Abstract < Full text). On the whole, we find that there has been a lack of studies to evaluate the effectiveness of graphical displays of document surrogates as indicators of relevance. This is mainly due to the fact that only recently has it been technologically feasible to render such displays in real-time by the computer. Our study is an attempt to fill that gap.

4 Experimental Setup

In this section, we discuss an experiment to test the effectiveness of the visualization tool as a first stage display, and as a tool to aid effective query reformulation. The part on query reformulation will be discussed in a subsequent paper. We used a portion of the TREC database consisting of all of disk1 and disk2 except the "Federal Register" documents. We used INQUERY 2.1p3 as the search engine [CCH92]. The retrieval mechanism of the search engine is based on bayesian inference networks using the

work occurrence statistics in documents. All of the TREC information topics that we used were very detailed in their description of information need. We picked ten information topics for this study.

The criterion used to pick the topics will be discussed below. A slightly modified version of the *Description* field (mainly removing the introductory words such as "Document will report") was submitted to the retrieval system. 120 documents from the top 150 retrieved documents were obtained and split into two groups as follows: High precision group consisting of 60 documents ranked 1 through 60 and a low precision group consisting of 60 documents ranked 91 through 150. We controlled for precision as a factor in the experiment since we felt that precision might impact the perusal time: Users might more quickly identify non-relevant documents, than the relevant documents. Also, earlier studies [Sar69, RRS61, MKB78] indicate that precision also influences the ability to judge non-relevance. Each of the two precision groups were further split into two groups: documents with odd ranks and the documents with even ranks. Thus, there were 4 groups of 30 documents for each information topic: High_precision_even_ranks, High_precision_odd_ranks, Low_precision_even_ranks and Low_precision_odd_ranks. The criterion used to pick the information topics for this study was that the "description" field when used as the query statement must retrieve a set of documents that had a distinct split in the precision values between the high precision group (ranks 1 through 60) and the low precision group (ranks 90 through 150). The precision values in the high precision group for all the chosen topics ranged from 0.43 to 0.6 while those of the low precision group ranged from 0.03 to 0.23.

The experiment we describe was aimed at investigating the effect of visualization on two problems for users:

- accurately identifying relevant documents
- effectively reformulating queries

In this paper, we report on results relevant to only the first of these, but because both problems were addressed in the same experimental design, we describe the entire experiment.

In the experiment, users were set two different types of tasks:

- Task of judging relevance: The users were given the information topic and the search statement used to retrieve documents. They were asked to judge the relevance of each of the 30 documents that were displayed to them as one of
  - relevant to the information topic.
  - non-relevant to the information topic.
  - Unsure.

  For the purposes of the current experiment, clicking the left mouse-button over a document title in the title-display window or over a vertical column in the visualization window marks the document as relevant. Clicking the right mouse button over the title (or the column in the visualization window) marks the document as non-relevant. Middle-clicking it marks the document as "Unsure". Also, left-clicking a query word in the visualization window marks all documents containing that query word as relevant. Right-clicking a query word marks all documents that do not contain that word as non-relevant. Full text or any other information about the documents was not made available to users.

- Query reformulation task: Here the users were asked to "modify the preconstructed query into a form that will retrieve more relevant documents". For half of the topics, users had the visualization tool and for the other half users did not have the visualization tool – making it a within-subjects, between-topics study.

For the "relevance judgment" task, precision (two levels: high and low) and visualization (two levels: with or without) were controlled in this within-subjects, within-topics study. The even ranked document group was shown with the visualization tool and the odd ranked document group was shown without the visualization tool. The users were told that the query was issued against 4 different databases and the top 30 documents from each database was presented to them as 4 separate tasks – two with and the other two without visualization. For a given topic, the first task was always a "relevance judgment" task with a high-precision group. The next task was a query reformulation task. The third, fourth and fifth tasks were relevance judgment tasks for the other three groups of 30 documents. The first task was always a relevance judgment task because we wanted the users to have a good feel for the retrieved set of documents before they embarked on the query reformulation task. The first task of relevance judgment was always done with a high-precision document group because, in the real-world the users almost always inspect the top-ranked high-precision document range before they go down the ranks to inspect the low-precision range. Each user did the 5 tasks (4 relevance judgment tasks for the 4 document groups, and one query reformulation task) for 6 information topics, and finally did the search reformulation task for 4 more topics. The 6 topics for which the users did both the relevance judgment and query reformulation were:

- Topic 77: Document will report a poaching method used against a certain type of wildlife.
- Topic 115: Document will report specific consequence(s) of the U.S.'s Immigration Reform and Control Act of 1986.
- Topic 134: Document will report on the objectives, processes, and organization of the human genome project.
- Topic 136: Document will report on attempts by Pacific Telesis to diversify beyond its basic business of providing local telephone service.
- Topic 145: Document will describe how, and how effectively, the so-called "pro-Israel lobby" operates in the United States.
- Topic 197: Document will discuss legal tort reform (a civil wrong for which the injured party seeks a judgment) with regard to placing limitations on monetary compensation to plaintiffs.

The order in which the six topics were presented were balanced across the 37 subjects. The order in which the two visualization conditions appeared for a given topic were also balanced. The order in which the two precision groups appeared in a given topic was not balanced due to the constraint that a high precision group is always the first condition.

The human subjects in this experiment were Georgia Tech undergraduate students enrolled in a one-credit hour class on library searching. Students who participated in the study got full scores in two homework assignments. The complete experiment was split over two days. Subjects were asked to sign a consent form upon arrival. They

were then given a demo of the system by the experimenter. They then had a hands-on tutorial where they practiced both the "relevance judgment" task and the "query reformulation" task. Then, they did the 5 tasks for each of the three information topics marking the end of the experiment for the first day. On the second day, they did the 5 tasks for each of the other 3 topics, followed by the "query reformulation" task for 4 other topics.

The subjects were given monetary incentive to do well in the experiment. They were evaluated as follows: We knew a-priori, the relevance of all the documents as given by the TREC assessors. For the relevance judgment task, for each document the user obtained a +1 point if their relevance judgment matches the TREC assessor's judgment, a -1 point if their judgment does not match, and 0 points if they are "Unsure". The user has to judge all of the 30 displayed documents. Thus, for the 4 groups of 30 documents, for the 6 topics, each subject made a total of 4x30x6 = 720 judgments.

		TREC	
		Rel	Not_rel
User	Rel	RR	RN
	Not_rel	NR	NN
	Unsure	UR	UN

The time taken by the subject to complete a task was also noted down. The top 10 quickest subjects with the most points were given monetary awards as follows: All participants were ranked on increasing order of time and decreasing order of points scored. Each participant's rank on both the categories (time and points) were added to get the sum-rank. The participant with the lowest sum rank was considered the best performer. Hence, to do well, one must be both accurate and quick. The top performer was given $50, the second and third performers were given $30 each, the fourth through sixth performers were given $20 each and the seventh through

the tenth performers were given $10 each. The participants were told of the rating scheme, so they optimized for time and accuracy equally.

Since we claim that graphical display of additional document surrogates does not increase perusal time significantly (due to the set-at-a-time perusal of documents), we predict that the time taken to complete the task for the visualization group will not be significantly higher than the non-visualization group. We also predict an increase in accuracy of relevance judgments for the visualization group, because we claim that very pertinent document surrogate information (i.e., the weight of query words in the retrieved documents) is being displayed in addition to the standard text surrogates such as title and source.

Effectiveness of the visualization tool was measured by what the subjects optimized upon: time, accuracy and the combined time_accuracy rank, where accuracy is the number of correct judgments minus the number of incorrect judgments after discarding the Unsure judgments, i.e., Accuracy = RR+NN-RN-NR. However, since the accuracy measure includes the correct judgments, Type I errors and Type II errors all in one score, we split the accuracy measure into distinct components. We define "Interactive Recall" as the ratio of the truly relevant documents that were judged as relevant by the user (i.e., Interactive Recall = RR/(RR + NR + UR)). We define "Interactive Precision" as the ratio of the documents judged as relevant by the user that are truly relevant (Interactive precision = RR/(RR + RN)). Here, a "truly relevant" document is a document that was judged relevant by the TREC assessor. Thus, if are trying to build an effective first stage display mechanism, we would strive for a display mechanism which would enable a user to pick (and read the full-text of) all of the relevant documents and only the relevant documents displayed. Picking

a non-relevant document as relevant would be time and money wasted perusing a non-relevant document. And, not being able to pick a relevant document, would be a missing out on relevant information.

However, "Unsure" documents pose a problem. It can be handled in two ways: If we assume that a user always reads the full text of an Unsure document, we should treat the Unsure documents as being judged relevant by the user. Conversely, if a user always skips over an Unsure document, we should treat the Unsure document as being judged non-relevant by the user. Below, we present the analysis with both the interpretations. Thus, if we assume the user to inspect the Unsure documents, we treat the Unsure documents as relevant.

Interactive Recall = (RR + UR) / (RR + NR + UR)

Interactive Precision = (RR + UR) / (RR + UR + RN + UN)

If we assume the user to not inspect the Unsure documents, we treat the Unsure documents as not-relevant,

Interactive Recall = RR / (RR + NR + UR)

Interactive Precision = RR / (RR + RN)

5 Results

Since there were 37 subjects, and all subjects did 6 topics with 4 tasks (for each of the 4 groups within the topic) per topic, there were a total of 37 x 6 x 4 = 888 observations. The approach used in all analyses was to construct a least squares, linear additive model of each performance measure as a function of the main effects

and interactions of the manipulated experimental variables.

The need for consideration of possible learning/ordering effects, due to the same subjects providing multiple responses at various experimental conditions, is minimized by the balancing of the order in which different experimental conditions are presented to the subjects. However, due to the requirement that within a topic, the high precision condition always be presented first, this balance could not be achieved for this factor. To account for this, the model included a term representing the observation order within subject/topic combination. The design thus allows for independent estimation of all effects except precision and observation order. The analysis presented will focus on the statistical significance of each term assuming the presence of the the other term in the model (i.e on the adjusted sums of squares in the Analysis of Variance (ANOVA) tables, as this provides evaluation of the marginal effect.

The residuals of the models constructed were analyzed to assure reasonable compliance with the normality, independence and constant variance assumptions required for validity of ANOVA,

The ANOVA tables for $log_{10}(time)$, accuracy and final score are shown in Tables X, XX and XXX. The means and standard errors are shown in table Z. As can be seen from the tables, viz is significantly better than noviz for logtime, accuracy and final score. It is also clear that low precision condition does significantly better than high precision for logtime, accuracy and final score. The interaction effects of precision and visualization are shown in figures 2, 3 and 4 with a 95% confidence interval around the means. When precision is high, visualization does not significantly affect logtime, but there is a decrease in logtime of 0.08 when precision is low. This corresponds to a reduction of 17.2 seconds, nearly a 20% decrease in average time required. Thus we

can conclude that the visualization tool helps users in identifying document relevance *more quickly.*

However, for the accuracy measure, there is no significant interaction between precision and visualization as shown by the almost-parallel lines in figure 3. Precision has a huge impact on accuracy, again consistent with previous studies [Sar69, MKB78]. While the effect of visualization on accuracy is significant, it is not as huge as the effect of precision. Users can identify document relevance *more accurately* with the visualization tool than without. The ability of users to identify non-relevant documents as non-relevant is much higher than their ability to identify relevant documents as relevant. This is reflected in the significantly very high accuracy value for low precision than for high precision.

Final score is a rank measure, which reflects the users ability to *accurately and quickly* identify document relevance. It is plotted in figure 4. Lower values are better for final score. As with accuracy, precision has a much higher impact than visualization, but both variables have a significant effect. Visualization tool helps and so does low precision. The improvement in final score averages 40.8 when using the visualization tool.

Table X. Analysis of Variance for logtime

Source	DF	Seq SS	Adj SS	Adj MS	F	P
precis	1	0.91889	0.45954	0.45954	30.56	0.000
viz	1	0.36761	0.36761	0.36761	24.44	0.000
precis*viz	1	0.29215	0.29215	0.29215	19.43	0.000

Table XX Analysis of Variance for Accuracy

Source	DF	Seq SS	Adj SS	Adj MS	F	P
precis	1	16782.27	11842.00	11842.00	532.55	0.000
viz	1	490.54	490.54	490.54	22.06	0.000
precis*viz	1	24.67	24.67	24.67	1.11	0.293

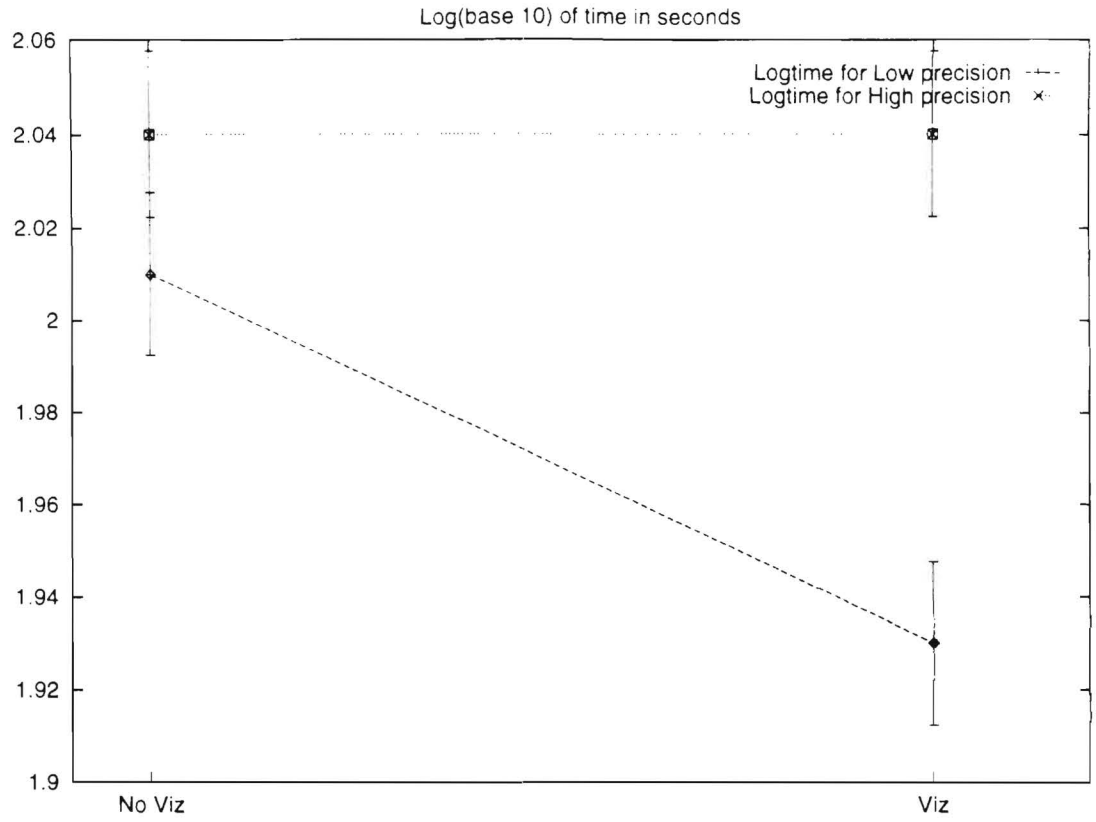


Figure 2: Interaction effects of precision and visualization on logtime.

Table XXX. Analysis of Variance for Final Score

Source	DF	Seq SS	Adj SS	Adj MS	F	P
precis	1	9901156	6789177	6789177	429.68	0.000
viz	1	362841	362841	362841	22.96	0.000
precis*viz	1	133133	133133	133133	8.43	0.004

TABLE Z. Least Square Means and Standard errors for Logtime, Accuracy and Final score (independent variables: precision 0-High, 1-Low and visualization 0-present, 1-absent):

PRECIS	VIZ	LOGTIME	ACCUR	FINSCOR
0	0	2.01	15.72	353.2
0	1	1.93	17.54	288.4
1	0	2.04	5.72	576.1
1	1	2.04	6.87	560.2
STD ERR OF EST		0.009	0.35	9.4

As discussed before, accuracy combines the following four items into one: ability to judge relevant and non-relevant documents (RR + NN), type I error, i.e., wrongly

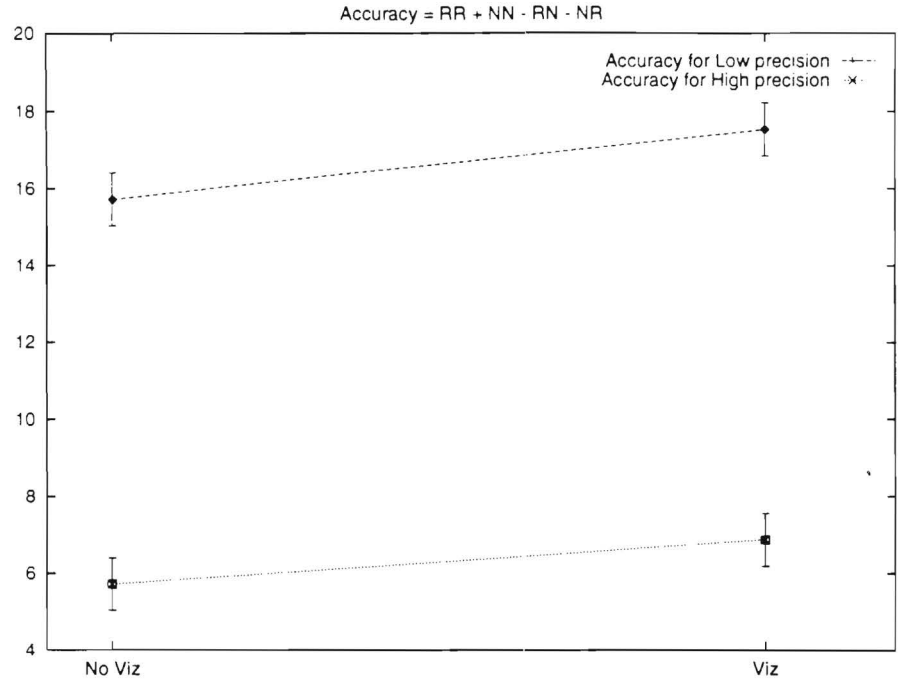


Figure 3: Interaction effects of precision and visualization on accuracy.

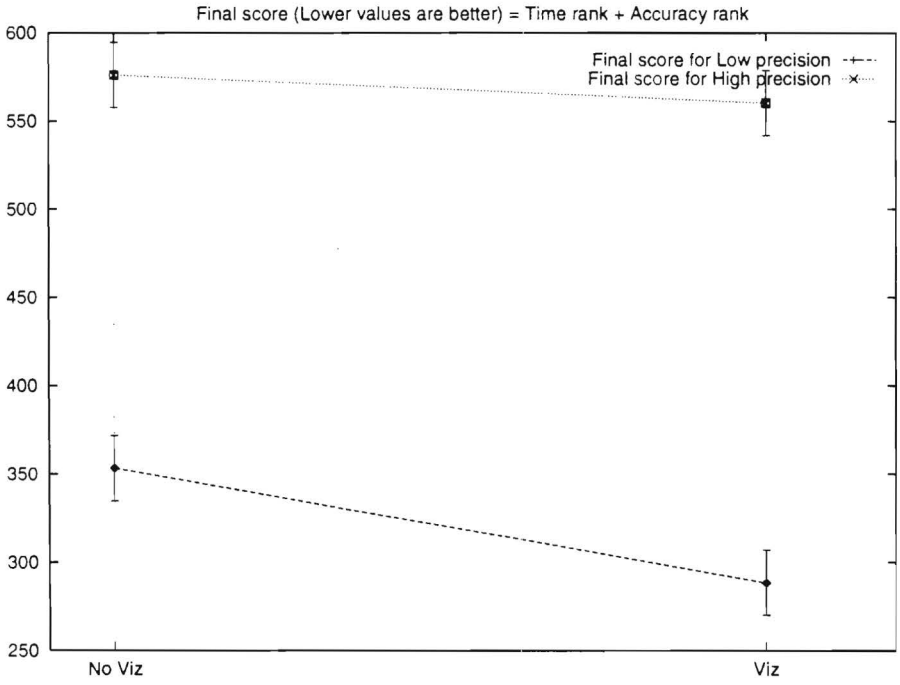


Figure 4: Interaction effects of precision and visualization on Final Score.

rejecting relevant documents, and type II error, i.e., wrongly accepting non-relevant documents. We feel that identifying non-relevant documents (NN) in and of itself is not as important as the other 3 items. For, it is important

- to minimize Type I errors, or else one runs the risk of missing out too many relevant documents.
- to minimize type II errors, or else one runs the risk of wasting too much money and effort in examining non-relevant documents.

We can capture all the interesting data with interactive recall and interactive precision as described in the previous section. In our tables, the interactive precision and interactive recall are denoted by "iprecwu" and "irecwu" respectively when users are assumed to treat unsure documents as relevant. Correspondingly, when unsure documents are assumed to be treated as non-relevant, interactive precision and interactive recall are denoted by the mnemonics "iprecwou" and "irecwou" respectively.

In considering the interactive precision measure there are a large number of cases where the values result in responses of zero divided by zero when users did not pick any of the displayed documents as relevant. Rather than eliminate these cases, the raw data (i.e., RR, RN, NR, NN, UR, UN) was aggregated over high and low precision levels for the same viz condition and the interactive precision measures then computed. Thus, for example, for topic 77, the RR values for the high_precision_viz case for subject 1 was added to the RR value of the low_precision_viz case of the same subject 1 and same topic 77. Now we end up with 444 observations instead of the original 888 observations. This eliminated the need for the "precision" term in the model, although the variability due to this factor is included in the error term.

However for iprecwou, there remain 2 cases where the response variable is still zero divided by zero. The result is a design where estimated effects are minimally dependent. Also, there are some quantization errors introduced in the interactive precision measure due to the denominator value being too close to zero[2]. The statistical significance of visualization for iprecwou, iprecwu, irecwou and irecwu are shown in table X1, and table Z2 shows the estimated means.

Visualization had no significant effect on interactive precision when Unsure documents were treated as non-relevant (iprecwou) at the 0.05 level, however, it was significant when Unsure documents were treated as relevant (iprecwu) (See figure 5). Although statistically significant, the absolute increase in interactive precision is very minimal (about 0.015). However, visualization had a significant effect on interactive recall (both irecwou and irecwu). Also, in the absolute sense, the improvement in interactive recall due to visualization is approximately 0.07 +/- 0.02. Clearly this is of sufficient magnitude to be of practical importance.

TABLE X1 ANOVA for iprecwou, iprecwu, irecwou and irecwu showing the effect of visualization

Metric	DF	Seq SS	Adj SS	Adj MS	F	P
iprecwou	1	0.03245	0.03065	0.03065	3.08	0.081
iprecwu	1	0.04364	0.04166	0.04166	5.45	0.021
irecwou	1	0.62496	0.62601	0.62601	35.89	0.000
irecwu	1	0.42259	0.42787	0.42787	27.46	0.000

TABLE Z2. Least squares means of iprecwou, iprecwu, irecwou, irecwu for visualization:

viz	iprecwou	iprecwu	irecwou	irecwu
0	0.6117	0.5753	0.4454	.5484
1	0.6284	0.5947	0.5209	.6108
Std error	.007	.006	.009	.008

[2]For iprecwou, there were 2 cases where the denominator had a value of 1, 5 cases of value 2, 6 cases of value 3. For iprecwu, there were 0 cases of denominator values 0 and 3, 1 case of values 1 and 2. Given that there were 444 observation points, these quantization errors are not expected to distort the results much.

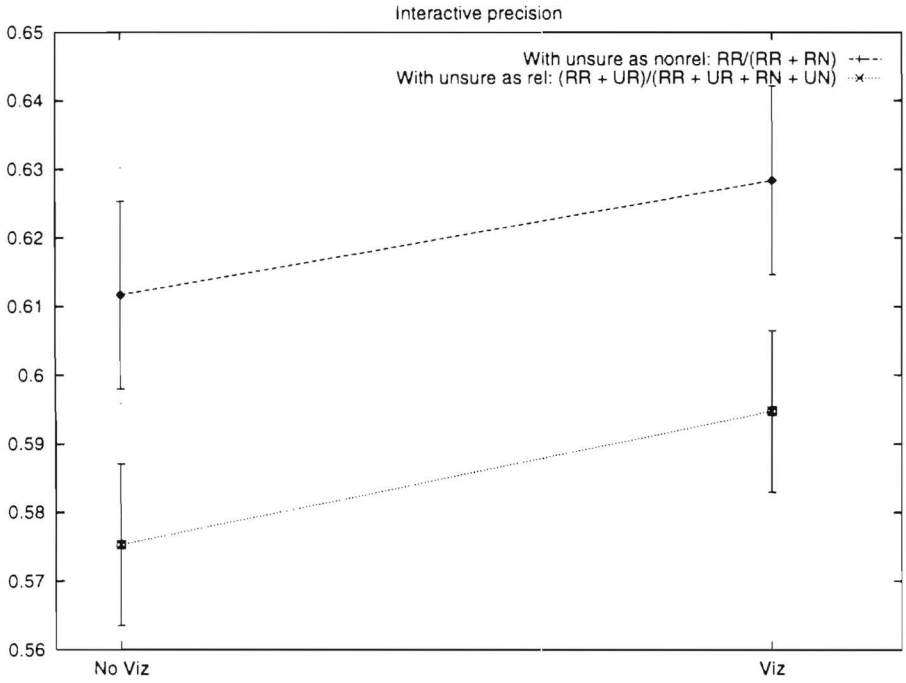


Figure 5: Effect of visualization on interactive precision (when Unsure documents are treated as relevant and non-relevant documents).

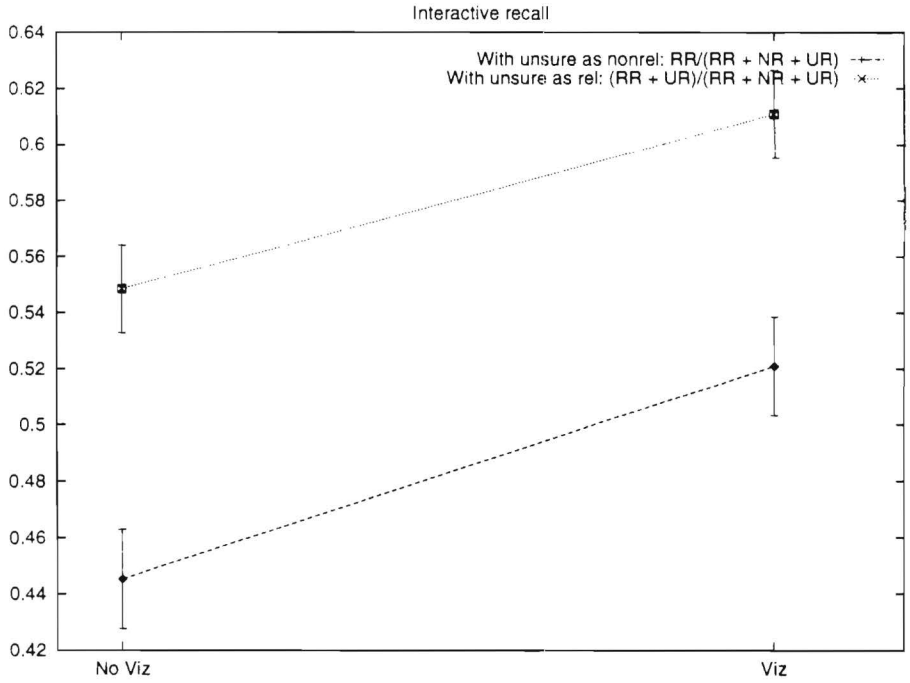


Figure 6: Effect of visualization on interactive recall (when Unsure documents are treated as relevant and non-relevant documents).

6 Conclusions

We have presented a visualization tool designed to be an effective first stage display of retrieved documents. User experiments empirically show that when precision is low, the visualization tool helps users in identifying document relevance **quicker** by about 20%. This is made possible due to the set-at-a-time perusal of graphical displays rather than document-at-a-time perusal of textual displays. The experiment also shows that users with the visualization tool did significantly better in **accurate** identification of document relevance. We broke down the accuracy measure into two components: interactive precision and interactive recall to gain a better understanding of the relevance judgment process. While the effect of visualization tool was marginally significant for interactive precision, it was highly significant for interactive recall. Thus, we can safely say that the visualization tool helps users in identifying more relevant documents out of the displayed documents. It also helps users in identifying them more quickly.

In an earlier paper [VB96], we discussed the difficulty of conducting interactive user experiments in IR. We mentioned the difficulty of huge inter-topic differences, inter-subject differences, the large number of subjects needed to account for these differences and how these factors severely affect the interactive track of TREC participants. There was also the problem of using appropriate measures to evaluate different user interface components and the lack of established metrics for these purposes. It is worthwhile noting how we approached these problems in the experiment described in this paper. At the outset, we had to be extremely specific in our claims about where the visualization tool would be of help. Having narrowed the scope

of the experiment to these claims, we had to devise a scheme where inter-subject and inter-topic variability could be kept to a minimum. By restricting the task to relevance judgment of documents, we could safely construct a within-topic, within-subject experiment that would not threaten the extensibility of our inferences to the real world.

In addition, in the absence of established interactive metrics, we had to come up with our own measures of effectiveness of graphical displays (such as interactive precision and interactive recall). It remains to be seen if such choice of metrics are appropriate and if they are of real impact in terms of the quality of interaction of end-users. The lack of convincing answers to the above questions points to the acute need for more interactive experiments to study human interaction with ranked output IR systems and to study the effectiveness of emerging display mechanisms such as visualizations.

Acknowledgements

We deeply appreciate the help of Neff Walker who helped us in the design of the experiment. Many thanks to Nick Belkin who has been constantly supportive of the work and providing valuable feedback to revisions of this paper. Support from ARPA contract No. F33615-93-1-1338 to the first author is appreciated.

References

[AB93] H.C. Arents and W.F.L. Bogaerts. Concept-based retrieval of hypermedia information – from term indexing to semantic hyperindexing. *Information Processing Management*, 29:387–396, 1993.

[ACRS93] M. Aboud, C. Chrisment, R. Razouk, and F. Sedes. Querying a hypertext information retrieval system by the use of classification. *Information Processing Management*, 29:387–396, 1993.

[BOB82] N.J. Belkin, R.N. Oddy, and H.M. Brooks. Ask for information retrieval: Parts I and II. *Journal of Documentation*, 38(2,3), 1982.

[CCH92] J.P. Callan, W.B. Croft, and S.M. Harding. The inquery retrieval system. In *Third International Conference on Database and Expert Systems Applications*, September 1992.

[CRM91] S. Card, G. Robertson, and J. Mackinlay. The information visualizer, an information workspace. In *Proceedings of CHI 91 Human Factors in Computer Systems.*, 1991.

[Hea95] Marti A. Hearst. Tilebars: Visualization of term distribution information in full text information access. In *Proceedings of CHI 95, Denver, Colarado.*, 1995.

[HKW94] Matthias Hemmje, Clemens Kunkel, and Alexander Willet. Lyberworld - a visualization user interface supporting full text retrieval. In *Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, pages 249–259, 1994.

[Jan91] Joseph W. Janes. Relevance judgements and the incremental presentation of document representations. *IPM*, 27(6):629–646, 1991.

[Kor91] Robert Korfhage. To see, or not to see - is that the query? In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 134–141, 1991.

[MFH95] Sougata Mukherjea, James. Foley, and Scott Hudson. Visualizing complex hypermedia networks through multiple hierarchical views. In *ACM SIGCHI*, 1995.

[MKB78] Richard S. Marcus, Peter Kugel, and Alan R. Benenfeld. Catalog information and text as indicators of relevance. *JASIS*, pages 15–30, Jan 1978.

[RRS61] G.J. Rath, A. Resnick, and T.R. Savage. Comparisons of four types of lexical indicators of content. *American Documentation*, pages 126–130, April 1961.

[Sar69] Tefko Saracevic. Comparative effects of titles, abstracts and full texts on relevance judgements. In *Proceedings of the ASIS*, pages 293–299, 1969.

[Spo94] Anslem Spoerri. Infocrystal: A visual tool for information retrieval and management. In *Human Factors in Computing Systems CHI 94 Conference Companion*, pages 11–12, 1994.

[Tho73] C.W.N. Thompson. The functions of abstracts in the initial screening of technical documents by the user. *JASIS*, 24:270–276, 1973.

[VB96] Aravindan Veerasamy and Nick Belkin. Evaluation of a tool for visualization of information retrieval results. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996.

PART 3

Metadata Management for Intelligent Query Processing

PUBLICATIONS (PART3):

[2.1]. Jeff Pittges. " Maintaining Instance-Based Constraints for Semantic Query Optimization," In *Proceedings of the Sixth IFIP TC-2 Working Conference on Data Semantics (DS-6)* , Stone Mountain, Georgia, May 1995

[2.2]. " Maintaining Semantic and Structural Metadata in the Metadata View Graph," J. Pittges, L. Mark, and S. Navathe. In *Proceedings of the Seventh International Conference On Management of Data*, Pune, India, December 1995.

Maintaining Instance-Based Constraints for Semantic Query Optimization

Jeff Pittges

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
(404) 853-9381
pittges@cc.gatech.edu
http://www.cc.gatech.edu/grads/p/Jeff.Pittges/pittges.html

May 3, 1995

Abstract

Semantic Query Optimization has traditionally relied upon scheme-based integrity constraints that are valid for all instances of a database. *Instance-based constraints*, which are only valid for certain states of a database, contain more information than scheme-based constraints because they are specific to the current contents of the database. This makes instance-based constraints more useful to semantic query optimization. However, instance-based constraints are highly sensitive to any changes made to the database and must therefore be updated and validated before they can be applied.

A Metadata View Graph (MVG) is a metadatabase that stores instance-based constraints, along with statistical and structural metadata, for logical views of the database. Constraints at this level are even more useful to semantic query optimization because they are specifically tailored to the intermediate results of a query. This paper reviews existing methods for constraint discovery, describes how constraints are stored in the Metadata View Graph at compile-time, and describes how the MVG Framework retrieves and maintains instance-based constraints at run-time. The paper then analyzes how to apply updates to instance-based constraints in order to refresh them.

Keywords: Metadata View Graph, Instance-Based Constraints, Semantic Query Optimization, Constraint Discovery, Metadata Maintenance.

1 Introduction

Semantic query optimization [HZ80, Kin81, SO89, CGM90, SSD92, HK93] uses transformation rules to reformulate a query into a semantically equivalent query that is more efficient to execute. Traditionally, transformation rules have been derived exclusively from scheme-based integrity constraints that are valid for all instances of a database. Transformation rules based on this type of constraint are desirable because the rules remain valid when changes are made to the database since changes cannot violate the integrity constraints. Unfortunately, scheme-based constraints are typically so general that they are of little use. For example, an integrity constraint may require an employee's salary to be greater than zero.

Recently, a number of researchers [YS89, SSS92, SHKC93, HK94] have proposed methods for discovering *instance-based constraints* (also referred to as *dynamic constraints* in [YS89] and *derived constraints* in [SSS92]) which are only valid for particular instances of the database. Instance-based constraints contain more information than scheme-based constraints because they represent the actual contents of the database. For example, an instance-based constraint may assert that the salary of all employees is greater than or equal to $22,000 and less than or equal to $85,000. Although this constraint is more useful than the integrity constraint given above, instance-based constraints are sensitive to changes made to the database, so they must be maintained whenever the database is updated.

A Metadata View Graph is a metadatabase that maintains instance-based constraints for logical views of the database. Constraints at this level are even more useful to semantic query optimization because they are specifically tailored to the intermediate results of a query. For example, one view might represent graduate students and another view might represent faculty. Both views would maintain an instance-based constraint on salary, but the salary range for graduate students would be much less than the salary range for faculty members. Therefore, when given a query involving graduate students, the semantic query optimizer could use the salary constraint for graduate students to reformulate the query.

In addition, capturing instance-based constraints for views of the database allows the query context to influence the optimization process during semantic query optimization. The *profitability* [SO87] of a rule can be adjusted for each view. Therefore, although the same rule may appear in several views, the rule can be applied differently based on the query context represented by each view. The rules for a particular view can be ordered by associating a context-sensitive salience (priority) with each rule. In this way, the views partition the semantic constraints which allows the constraints to be precisely tailored to particular data sets. Partitioning reduces the number of rules which must be searched during semantic query optimization and improves rule selection by prioritizing rules according to the query context.

The rest of this paper is organized as follows. Section 2 describes the Metadata View Graph Framework. This section provides a structural description of the Metadata View Graph, reviews existing methods for discovering instance-based constraints, and describes how instance-based constraints are stored in the Metadata View Graph. The section also

describes how instance-based constraints are retrieved at run-time and used to select the best query execution plan. Section 3 presents the general problem of maintaining instance-based constraints, presents various representations that allow instance-based constraints to be maintained, and analyzes the problem of applying updates to instance-based constraints in order to refresh them. The last section summarizes the contributions of this paper and describes future tasks for this research.

2 The MVG Framework

The Metadata View Graph Framework [PMN95] supports the integration of various approaches to query optimization. As shown in Figure 1, the framework consists of an optimizer and a metadatabase. The MVG Framework has been developed with two objectives in mind: (1) improve query optimization, and (2) provide for a highly extensible query optimizer. Query optimization is improved by maintaining metadata, especially instance-based constraints which improve semantic query optimization, multiple query optimization, incremental query computation, and dynamic plans. Three types of knowledge are required by the query optimizer: (1) procedural knowledge specific to each type of query optimization, (2) control knowledge which integrates the various types of query optimization together, and (3) domain knowledge (metadata about the database). The MVG Framework maintains these three types of knowledge separately in order to facilitate a highly extensible architecture. The optimizer and metadatabase can be extended incrementally as new approaches to query optimization are developed.

This research focuses primarily on the metadatabase (the Metadata View Graph) which was inspired by considering six types of query optimization: syntactic, physical, semantic, dynamic, multiple, and caching and incremental query computation. Therefore, we treat the query optimizer as a black box that is capable of performing these six types of query optimization. Within that black box, the optimizers can be developed independently and loosely coupled, which requires less integration effort but reduces run-time efficiency, or the optimizers can be tightly coupled, which requires greater integration effort (each optimizer may have to be rewritten) but improves run-time efficiency. Although we envision a set of rule-based optimizers [Fre87, GD87, GM93], the actual implementation is irrelevant to our work on Metadata View Graphs.

2.1 Metadata View Graphs

A Metadata View Graph (MVG) is a collection of networks, as shown in Figure 2, for organizing and storing metadata (i.e., a metadatabase). The Metadata View Graph consists of four components: (1) a lexicon, (2) a semantic network, (3) a view network, and (4) a QEP Network of query execution plans.

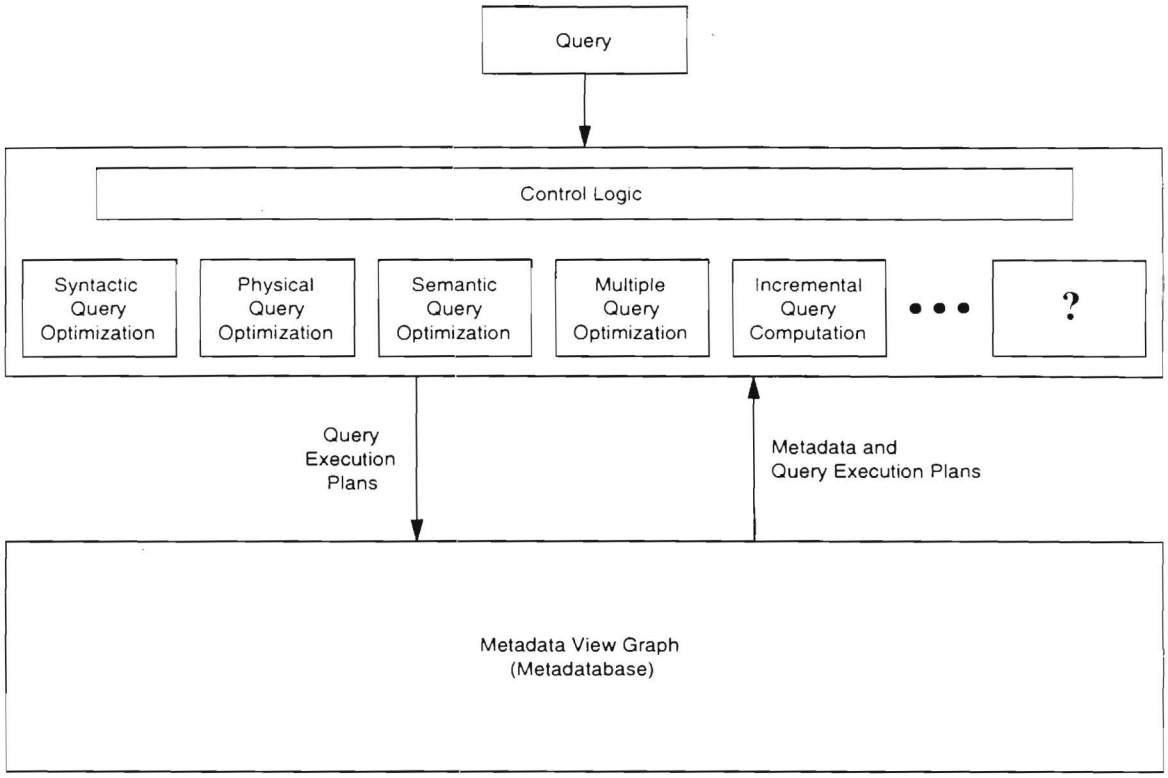


Figure 1: The MVG Framework.

The lexicon contains an entry for each term (word or phrase) recognized by the system (i.e., the system's vocabulary). A lexical entry provides information about the term, including a set of pointers to the semantic nodes that represent the term. In general, a lexicon will store any information about a term that is useful to the system.

Semantic networks represent domain knowledge about the concepts "understood" by the system. Each concept is represented as a node. Two nodes are linked together to represent their relationship to each other. In Figure 2, the semantic network is represented by nodes a_1 through a_8 and S_1 through S_4. Nodes a_1 through a_8 represent the attributes that participate in one or more of the base relations (nodes R_1 through R_3). The attribute nodes are linked directly to their corresponding base relations.

The attribute names and types are specified in the data definition of the database. These nodes form the foundation of the semantic network. The network can be extended by defining nodes and links for application specific concepts and relationships. For example, two nodes representing STUDENT and ADVISOR could be connected by the links ADVISED-BY and ADVISOR-OF. The lexicon and semantic network are not relevant to the research presented in this paper.

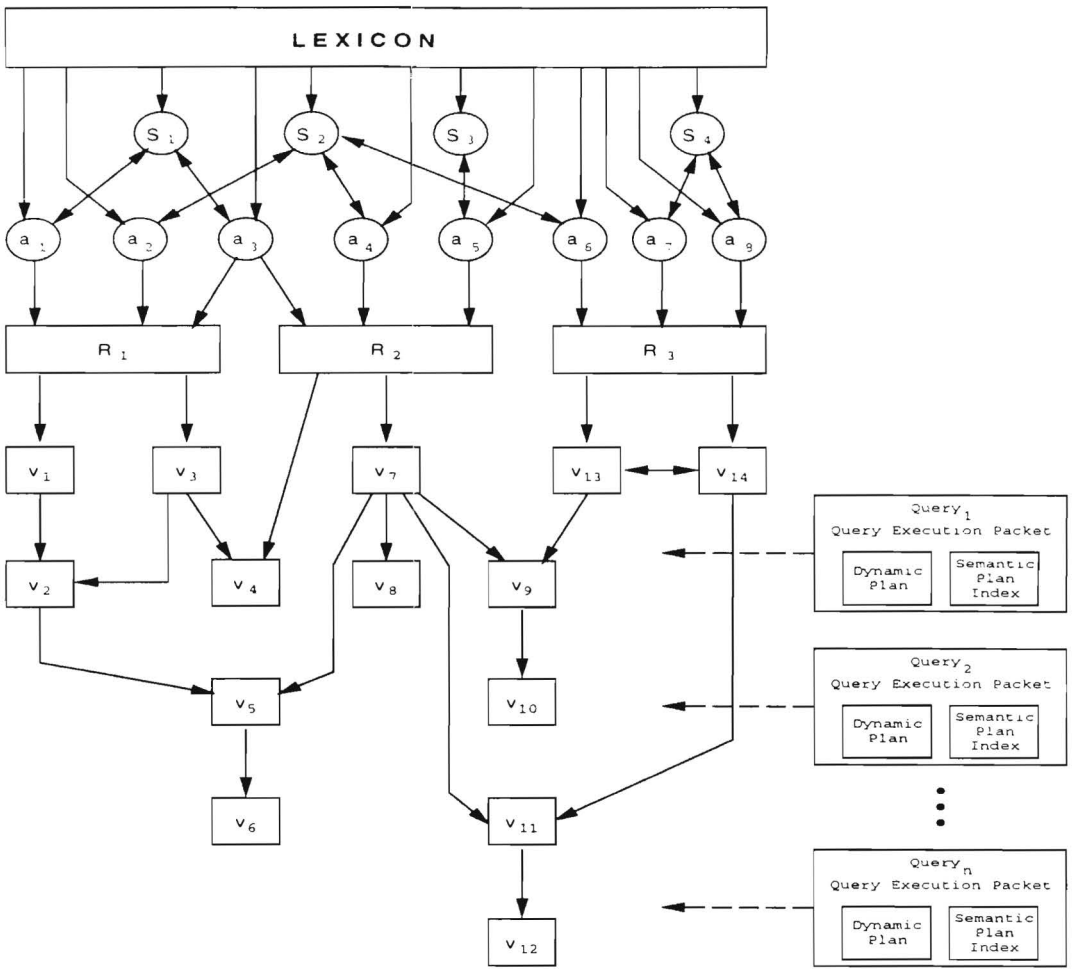


Figure 2: Conceptual Representation of a Metadata View Graph.

The View Network is an extension of Roussopoulos' Logical Access Path schema [Rou82]. The View Network stores semantic, statistical, and structural metadata that is useful to the query optimizer. The view nodes in the network, v_1 through v_{14}, represent *logical views* (intermediate results) and store metadata specific to the particular data set. A view is a projection of attributes which can be defined recursively as follows. All base relations are views. Additional views are the result of applying an operation (e.g., selection, projection, join) to a view or to a pair of views. The views represented by these nodes may or may not be materialized. The links represent logical operations and semantic relationships. The View Network, nodes R_1 through R_3 and v_1 through v_{14}, is essentially a collection of query graphs overlaid on top of each other where R_1 through R_3 represent base relations and v_1 through v_{14} represent the results of performing the operations specified by the links.

The View Network is a unified structure that applies to all of the application queries being served by the Metadata View Graph. The top level of the hierarchy consists of base relation nodes which anchor the Metadata View Graph and serve to connect the semantic network

to the View Network. The remaining nodes represent logical views. A view node is defined by the links connecting the node to the base relations. Figure 3 illustrates an example View Network along with a semantic network and lexicon.

The QEP Network stores two types of query execution plans, *dynamic plans* [GW89, CG94] and *semantic plans*. A semantic plan is a query execution plan that is semantically equivalent to the original query. Semantic plans are generated during semantic query optimization and depend on integrity constraints and instance-based semantic constraints.

A dynamic plan links several query execution plans together with choose-plan operators. An example of a dynamic plan is shown in Figure 4. Choose-plan operators allow a decision to be postponed until run-time when the run-time conditions are known. In order to select the best plan from the dynamic plan, the dynamic plan is traversed and the best path is chosen at each choose-plan operator. The statistics stored at the view nodes can guide the decision process. Therefore, the dynamic plan contains pointers to the view nodes with relevant metadata. If the statistics at a view node are out of date, the statistics must be updated before a decision can accurately be made.

The QEP Network maintains a separate *query execution packet* (i.e., a dynamic plan and semantic plan index) for each application query. Figure 4 illustrates how the query execution plans are linked to the logical access paths of the View Network so that the relevant view nodes can be retrieved for each plan. When a query is received at run-time, the query's execution packet is retrieved from the QEP Network and the metadata at the view nodes is used to select the best plan. When a semantic plan is selected as a candidate, the constraints it depends on must be updated and verified against the current state of the database before the plan can be executed. If one of the dependencies has been violated, then the plan is no longer guaranteed to be correct.

2.2 Using Metadata View Graphs

Metadata View Graphs are used at compile-time and run-time. Metadata is collected at compile-time and stored in the View Network. Metadata is retrieved at run-time and used to select the best query execution plan.

2.2.1 Compile-Time

The following high-level algorithm describes how the View Network and the QEP Network are constructed incrementally when a query is compiled.

1. When a query is received at compile-time, the query is optimized with conventional optimization techniques in order to generate a set of logical access paths.

2. The logical access paths are used to construct a separate View Network for the query being compiled. The existing MVG View Network is searched for (partial) matching view nodes.

3. The logical access paths are translated into query execution plans. The cost of each plan is estimated and the plans are filtered to remove any non-competitive plans.

4. Constraints and statistics are collected for the view nodes in the query's View Network. If a node already exists in the MVG View Network, metadata may not have to be collected for that node if the node's metadata is up to date.

   In order to collect metadata, each logical access path will have to be executed. If the query being compiled contains variables, the query history will be used to substitute values for the variables. These are the variable bindings most likely to occur in future queries.

5. The constraints collected in step 4 are used by the optimizer to generate additional query execution plans. Semantic query optimization, multiple query optimization, and incremental query computation can apply the instance-based constraints that were collected.

6. If a new set of query execution plans are produced in step 5, the plans are evaluated using the statistics that were collected. The non-competitive plans are discarded and the query's View Network is modified to include any additional view nodes. Steps 4 and 5 are repeated until no new query execution plans are generated.

7. A query execution packet is created for the query. A dynamic plan is constructed for the non-semantic query execution plans, and an index is created for the semantic plans. The dynamic plan and the semantic plan index are stored in the query's execution packet.

8. The query's View Network is unified with the MVG's View Network (i.e., if a view node in the query's View Network does not already exist in the MVG's View Network, the node is added to the MVG's View Network at the correct location and the MVG View Network is reorganized).

Constraint Discovery

Our research does not address constraint discovery. This section describes existing methods for constraint discovery and discusses the advantages provided by the Metadata View Graph Framework.

Two primary problems hinder constraint discovery: (1) determining *where* to search, and (2) determining *what* to search for. Focusing on views of the database reduces the search space and produces more useful constraints. Searching smaller data sets, as opposed to searching the entire database, improves the performance of the discovery methods. Therefore, the view nodes of the Metadata View Graph determine where to search.

Two basic methods are used to determine what to search for: (1) query-driven methods, and (2) data-driven methods. Query-driven methods use a top-down process to search for constraints that would have been useful for previous queries. Data-driven methods use a bottom-up process to search (random) data sets for constraints. Metadata View Graphs provide a framework for integrating the top-down and bottom-up processes.

Query Driven Methods

[SSS92] presents a query-driven method for discovering constraints. Given a query, the semantic query optimizer identifies the *template* transformation rules that would have been useful to the optimizer, and the system discovers constraints that fit the rule templates. This strategy reduces the search space by only considering data sets that are relevant to queries that have been received. The disadvantage of this strategy is that a query will only benefit from the constraints that have been discovered if the query is similar to a previous query.

Reverse Engineering Method

After a query has been executed, [YS89, HK94] inspect the query result and attempt to discover relationships with other queries. For example, if the (intermediate) results of two queries are identical, then there must be some constraints relating the two queries. This is a type of query-driven approach that requires two similar queries before any constraints are discovered. In addition, this method requires that the results of previous queries be stored and matched against future queries.

Data Driven Methods

[SHKC93] has proposed a data-driven approach that uses grid files to inspect combinations of attribute values for a given data set. The zeros in the grid file indicate constraints. The advantage of this approach is that constraints can be found regardless of the query history. However, since it is impractical to search the entire database, there is no guarantee that the discovered constraints will apply to a query.

MVG Guidelines

The View Network is constructed for the application queries that have been compiled by the system, thus providing queries for the query-driven methods. In addition, the View Network identifies relevant data sets for the data-driven methods. Therefore, the View Network provides a foundation for integrating the top-down and bottom-up constraint discovery processes.

The semantic query optimization transformation types should be used to guide the discovery process. This guideline will focus the data-driven methods on constraints that are useful given the current structure of the database.

For example, one transition type attempts to introduce an index into the query condition. Therefore, the indexed attributes of each base relation should be explored since constraints involving these attributes could lead to rules that introduce indexes. Another transformation

attempts to eliminate operations such as a join between two views. In some cases, range constraints for the join attributes of each view can determine that the result of a join is empty in which case the operation can be eliminated.

The structure of the View Network, which consists of chains of nodes organized in a subsumption hierarchy, can also be used to guide the data-driven techniques. The constraints that exist at higher level nodes can be propagated to the nodes below, provided they apply to the nodes below, and then tightened to reflect the contents of the more restricted view. At the top level, the scheme-based integrity constraints that already exist for a database can be restricted to reflect the actual contents of the base relations.

Run-Time

When a query is received at run-time, the query's execution packet (i.e., the query's dynamic plan and its semantic plan index) is retrieved from the QEP Network along with any relevant view nodes from the View Network. The semantic plans are indexed so that the run-time bindings of the query can be used to select the semantic plans that match the conditions of the query. A semantic plan contains a set of pointers to the instance-based constraints it depends on (i.e., the constraints used to generate the plan). These dependencies must be verified for the current state of the database before a semantic plan can be executed. If any of the constraints that a plan depends on are no longer valid, the semantic plan is not guaranteed to be correct.

Each intermediate result in a query execution plan indexes a (possibly empty) set of view nodes with relevant metadata (i.e., metadata that is useful for predicting statistics about the intermediate result). When plans are being compared, the statistics at the view nodes are used to estimate the cost of each plan. However, before the plans can be evaluated, the metadata must be refreshed to reflect any updates made to the database.

The query optimizer selects the best non-semantic plan from the query's dynamic plan. If it is cost effective to update the instance-based constraints, then the constraints are updated and the semantic plans are evaluated. The best query execution plan is selected for execution. The Metadata View Graph adds the query to its query log along with a time-stamp and any other data which may have been collected during execution of the query.

Although this scheme moves most of the optimization effort to the compile phase, it does not preclude run-time optimization. For example, if several queries are received within a reasonable time frame, multiple query optimization can be performed.

2.3 Selecting a Query Execution Plan at Run-Time

Consider the two base relations and the *template query*, Q_1, shown below. A template query contains one or more variables. Instantiations of a template query are received at run-time with all of the variables bound.

Relation	Attributes
Students	snum, class, GPA, advisor
Employees	enum, salary, dept, pos

Q_1: Select GPA
From Employees, Students
Where pos = student AND dept = var_1
AND enum = snum

Query Q_1 requests the grade point averages of the students in the var_1 department who are employeed. Figure 3 illustrates part of a view network that supports this query. In this example, there are four departments (Psychology, Math, Computer Science, and Business). The View Network is not required to contain a view node for every department. Only the most frequently accessed views, based on the query history, will be represented in the view network. For example, only three class views ($v_9 - v_{11}$) are represented for the Student base relation.

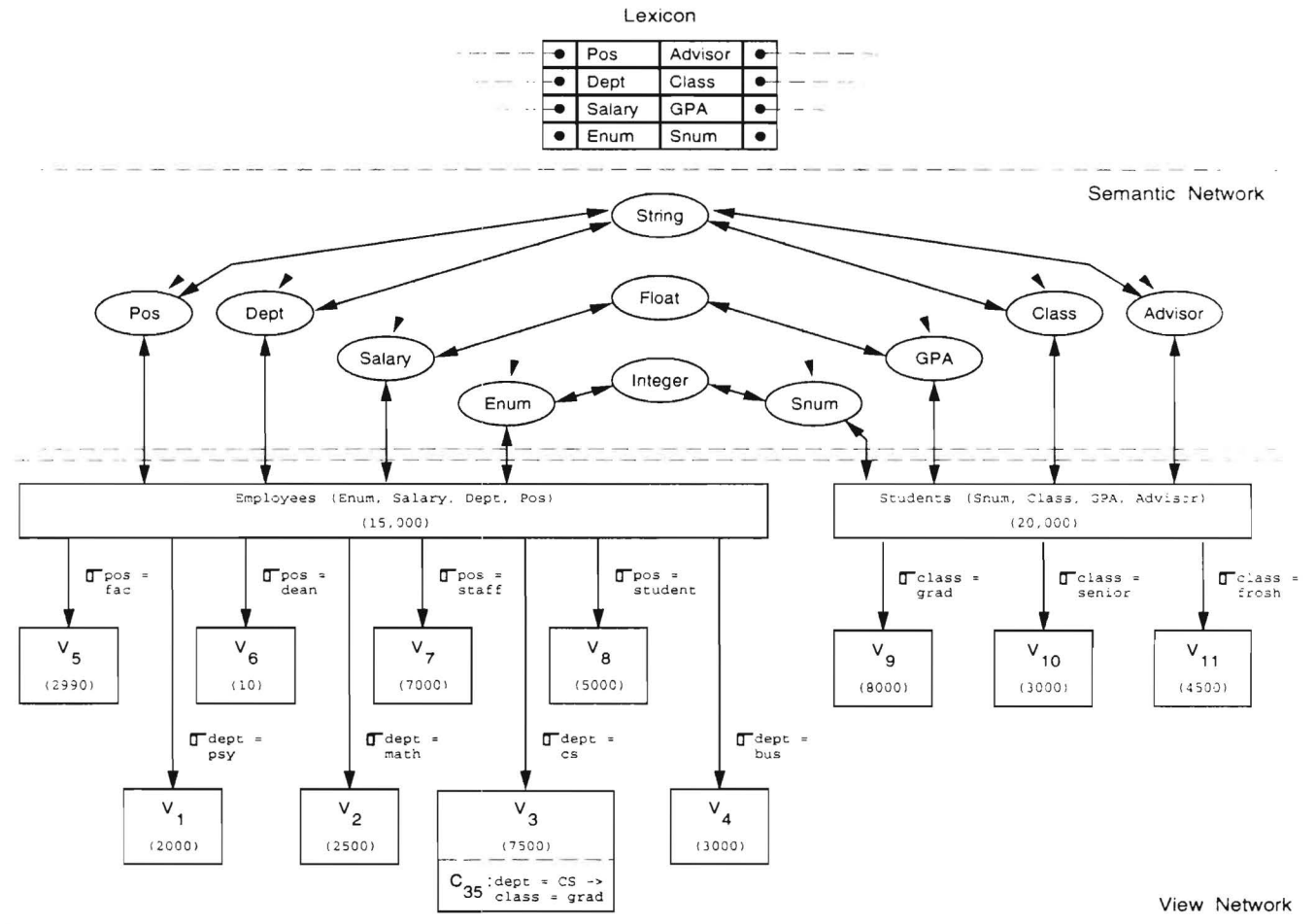


Figure 3: Example Lexicon, Semantic Network, and View Network.

Figure 4 shows the Query Execution Packet for query Q_1. The packet contains the dynamic plan and the semantic plan index. There is one choice to be made in the dynamic plan. The

two selections can be performed in either order. The first two filter operations point to the view nodes that contain statistics to determine which path to take. Once var_1 is bound at run-time, the choice is obvious. If DEPT = CS, then POS = STUDENT produces a smaller intermediate result (5000 tuples) than DEPT = CS (7500). For the other three departments, however, selecting the department first produces the smaller intermediate result.

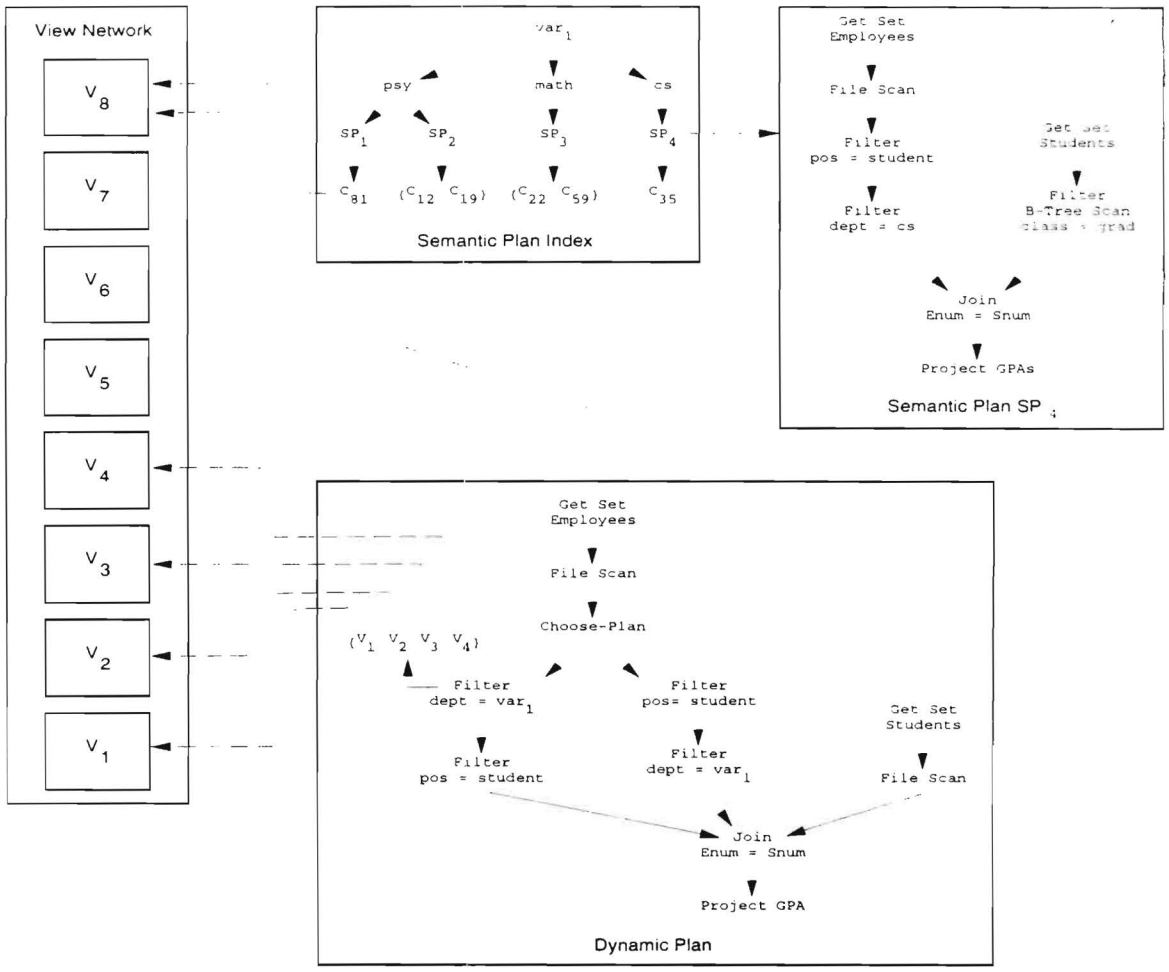


Figure 4: A global persepctive of selecting a query execution plan at run-time.

Semantic plans are indexed by the bindings of the variables. The packet contains four semantic plans, SP_1, SP_2, SP_3, and SP_4, corresponding to the variable bindings var_1 = PSY, var_1 = MATH, and var_1 = CS respectively. There are two semantic plans for the PSY binding. Each plan points to the set of constraints that the plan depends on.

Assume an instantiation of query Q_1 is received at run-time with var_1 bound to CS. The semantic plan index would return plan SP_4 which depends on constraint C_{35} which is stored at view node v_3. The semantic plan SP_4 is shown in Figure 4. Constraint C_{35}, shown in Figure 3, states that if the department is CS then the class must be GRAD. In other words, only graduate students from the computer science department are employed.

Assuming there is an index on the CLASS attribute of the Student base relation, the semantic plan SP_4 can use that index to reduce the size of the join between the Student base relation and the view of employed computer science students. Furthermore, because the semantic plan was designed for a specific binding of var_1, the choose-plan operator can be omitted since a drastic change in the database would be required before the number of students would be greater than the number of computer science employees.

Note that constraint C_{35} is not an integrity constraint. Undergraduate computer science students can be employed. Therefore, if the database were updated to include an employed undergraduate from the computer science department, the semantic plan SP_4 would no longer be valid. Consequently, the constraint C_{35} must be updated and verified against the current state of the database before the semantic plan can be selected for execution. If an update invalidates a constraint, then none of the semantic plans that depend on the invalid constraint can be executed because the plans are not guaranteed to be correct.

3 Maintaining Instance-Based Constraints

When a semantic plan is selected from the QEP Network, the constraints that the plan depends on must be refreshed and verified for the current state of the database. Constraints may be out of date if one or more updates have been received by the system since the last time the constraints were refreshed. However, only a subset of the updates will apply to a view node based on the definition of the node (e.g., GPA $\geq$ 3.5). Therefore, the updates must be *filtered* to remove the *irrelevant updates* (i.e., those updates that do not apply to the view node being updated) [BLT86]. An example is provided below.

Consider the Student base relation and the view node shown in Figure 5. The view node shown in this example contains seven constraints, a tuple count, a distribution profile, and a view cache pointer (since the pointer is nil, there is no view cache for this node). The view is defined for students with a GPA of 3.0 or greater. The constraints and tuple count at the view node can be verified by selecting the tuples from the base relation (as shown) that satisfy the definition of the node (i.e., GPA $\geq$ 3.0). An update contains a unique time-stamp, which indicates when the update was received, along with the tuple to be inserted or deleted. Updates to the database are maintained in a set of logs, one log per base relation. In order to refresh the metadata at this node, the updates in the base relation update log must be filtered to select the updates that satisfy the definition of the node. Each update with a GPA of 3.0 or greater can then be applied to each metadatum at the node.

Before the updates can be filtered and applied to the metadata, the update logs and the view nodes to be updated must be retrieved from disk. The cost of these disk accesses dominates the cost of the update process. Therefore, maintaining metadata can be divided into three subproblems: (1) managing the update logs, (2) managing the view nodes, and (3) refreshing the metadata at a view node. The rest of this paper analyzes the problem of refreshing constraints once the updates and the constraints are in main memory.

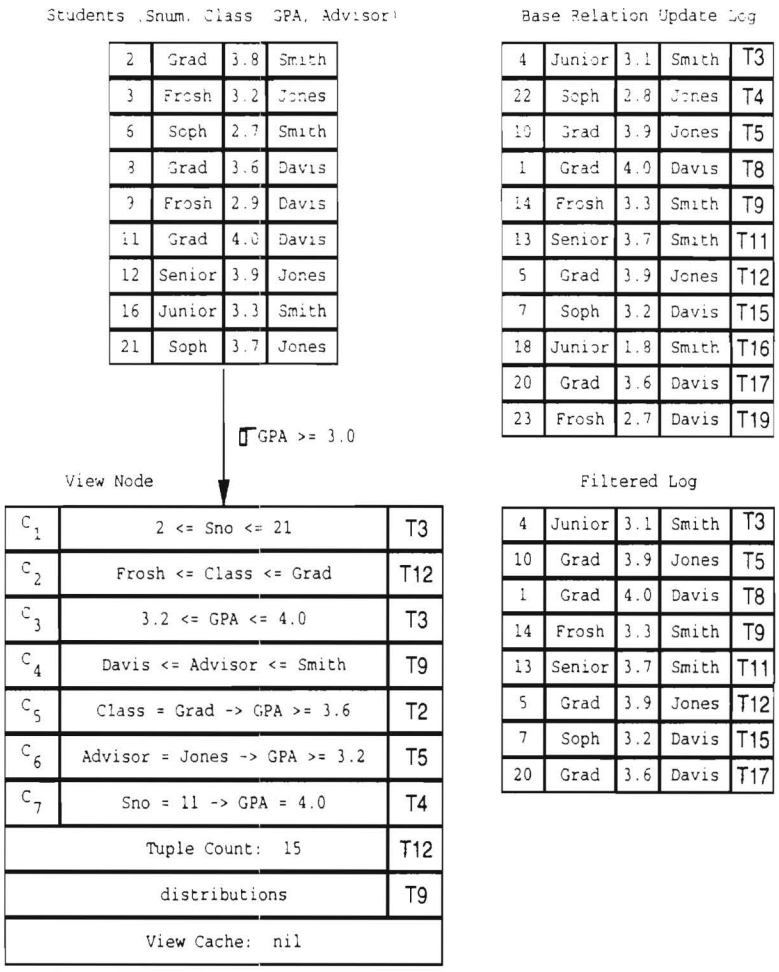


Figure 5: The Student base relation, base relation update log, view node, and filtered log.

3.1 Representing Constraints

Constraints are represented in First Order Predicate Logic. However, constraints must maintain additional information in order to be maintained efficiently. This section begins by considering the affects that insertions and deletions have on instance-based constraints, and then the section considers several representations that improve the maintainability of instance-based constraints.

3.1.1 Update Affects

The Metadata View Graph stores semantic query execution plans that depend on certain conditions. Consider a semantic plan, SP_1, that requires that all graduate students have a GPA greater than 3.0. If all of the graduate students in the database have a GPA of 3.2 or

greater, then the instance-based constraint, C_1: CLASS = GRAD $\Rightarrow$ GPA $\geq$ 3.2, indicates that the semantic plan SP_1 is valid.

Assume there is only one graduate student with a 3.2 GPA and assume that the student is deleted. The deletion does not have to be applied to C_1 because no deletion could violate the required condition, CLASS = GRAD $\Rightarrow$ GPA > 3.0. In general, unless a constraint implies existence, deletions do not have to be applied to valid constraints because deletions cannot invalidate the required conditions with respect to the semantic plans that are stored in the Metadata View Graph.

However, assume a graduate student with a 2.8 GPA is inserted into the database. The insertion must be applied to C_1 because the update violates the required condition for SP_1 thus invalidating the plan. Now assume that the graduate student with GPA 2.8 is deleted and assume that the remaining graduate students all have a GPA greater than 3.0. The required condition for SP_1 is now satisfied. Therefore, this deletion should be applied to C_1 in order to validate the plan.

When insertions are made to the database, it is easy to modify constraints because all of the necessary information is contained in the update. However, when deletions occur, the constraint must represent additional information in order to *recompute* the correct constraint.

3.1.2 Efficient Representations

Consider a student base relation with four attributes: student number (Snum), class (e.g., frosh, grad), GPA, and advisor. Consider a view node defined for GPA $\geq$ 3.9 and assume that constraint C_1 is stored at the view node. Assume that student 4 is a graduate student advised by Smith with a 3.92 GPA. If student 4 is inserted at time T_2, constraint C_1 can be modified as shown below. However, if student 4 is deleted at time T_3, the constraint cannot be recomputed.

T_1: C_1: Class = Grad $\Rightarrow$ Advisor = Jones

T_2: Insert: Snum = 4, Class = Grad, GPA = 3.92, Advisor = Smith

T_2: C_1: Class = Grad $\Rightarrow$ (Advisor = Jones) OR (Advisor = Smith)

T_3: Delete: Snum = 4, Class = Grad, GPA = 3.92, Advisor = Smith

T_3: C_1: Class = Grad $\Rightarrow$ Advisor = Jones

When student 4 is deleted, the constraint should be modified as shown at T_3. However, with this representation, the system must materialize the view (GPA $\geq$ 3.9) in order to discover that there are no graduate students in the view who are advised by Smith. We will consider two solutions to this problem.

1. Recompute the constraints at run-time during query execution by testing all of the tuples in the intermediate result. This can be done by saving the intermediate result, possibly to disk, and recomputing all of the constraints at a node, or by recomputing a few constraints on the fly while the query is being executed.

2. When the attribute of a term can be enumerated, as in the example above, keep a counter for each term.

The first solution requires additional processing during run-time to recompute the constraints. We can assume that the processing cost is negligible or that the processing is performed when the system is idle. However, the problem with this solution is that the constraints are recomputed during or after query execution. Therefore, the constraints cannot be used for the given query.

Consider semantic plan SP_1 which requires that all graduate students have a GPA greater than 3.0. Assume that SP_1 is valid at time T_1. At time T_2, a graduate student with a 2.8 GPA is inserted which invalidates semantic plan SP_1. At T_3, however, the graduate student with GPA 2.8 is deleted. At this point, semantic plan SP_1 is valid, but the plan cannot be used because constraint C_1 cannot be recomputed until the query is executed.

Therefore, the constraints will have to be recomputed every time a deletion affects the view node. Consequently, a constraint may thrash between being valid and invalid and the semantic plans that depend on the constraint will never be usable even though the required conditions are met.

The second solution is preferable, but this representation does not apply to attributes with non-enumerable values. Consider the following representation for constraint C_1. This representation maintains the number of tuples that apply to each condition. At time T_1, there are 10 graduate students in the view and all 10 are advised by Jones.

T_1: C_1: Class = Grad (10) $\Rightarrow$ Advisor = Jones (10)

T_2: Insert: Snum = 4, Class = Grad, GPA = 3.92, Advisor = Smith

T_2: C_1: Class = Grad (11) $\Rightarrow$ (Advisor = Jones (10)) OR (Advisor = Smith (1))

T_3: Delete: Snum = 5, Class = Grad, GPA = 3.94, Advisor = Jones

T_3: C_1: Class = Grad (10) $\Rightarrow$ (Advisor = Jones (9)) OR (Advisor = Smith (1))

T_4: Delete: Snum = 4, Class = Grad, GPA = 3.92, Advisor = Smith

T_4: C_1: Class = Grad (9) $\Rightarrow$ Advisor = Jones (9)

Student 4 is inserted at T_2 and the constraint is modified. There are now 11 graduate students in the view, 10 are advised by Jones and 1 is advised by Smith. Student 5 is deleted at T_3

and the constraint is modified accordingly. Once again there are 10 graduate students in the view, but now there are 9 students advised by Jones and 1 advised by Smith. Finally, student 4 is deleted at T_4. The constraint is modified to reflect that all 9 graduate students in the view are advised by Jones.

This representation works well for attributes with values that can be enumerated. Some constraints have attribute values that cannot be enumerated. For example, consider the range constraint on GPA shown below.

T_{10}: C_2: GPA >= 3.56

T_{11}: Insert: Snum = 4: GPA = 3.5

T_{11}: C_2: GPA >= 3.5

T_{12}: Delete: Snum = 4: GPA = 3.5

Initially, the lowest GPA for the given view is 3.56. When student 4 is inserted at time T_{11}, the lowest GPA becomes 3.5. However, when student 4 is deleted at T_{12}, there is no way to determine the exact value of the lowest GPA in the view without materializing the view. However, constraint C_2 still represents a lower bound on GPA. The constraint asserts that all of the GPAs are greater than or equal to 3.5, but the constraint does not represent the exact value of the lowest GPA.

The representation shown below maintains a *boundary value list* of the lowest GPAs. This representation can be used to determine the exact lower bound provided the boundary value list is not empty.

T_{10}: C_2: GPA >= (3.56, 3.57, 3.57, 3.58, 3.59)

T_{11}: Insert: Snum = 4: GPA = 3.5

T_{11}: C_2: GPA >= (3.5, 3.56, 3.57, 3.57, 3.58, 3.59)

T_{12}: Delete: Snum = 4: GPA = 3.5

T_{12}: C_2: GPA >= (3.56, 3.57, 3.57, 3.58, 3.59)

Initially the boundary value list contains the 5 lowest GPAs in the view. When student 4 is inserted at T_{11}, student 4's GPA is the lowest GPA in the view. Therefore, student 4's GPA is added to the front of the boundary value list. When student 4 is deleted at T_{12}, one instance of the value 3.5 (student 4's GPA) is removed from the boundary value list.

If the boundary value list becomes empty, because all of the values are deleted and no insertions replace them, then the view must be materialized in order to determine the exact lower bound. The last value in the boundary value list can be retained in order to provide a lower bound for the constraint. In this case, the retained value must be flagged as invalid so that it can be removed when the boundary value list is recomputed. If an insertion is received with a GPA less than or equal to the retained value, then the inserted GPA will replace the invalid GPA and the constraint will once again reflect the exact lowest GPA in the view.

For example, consider constraint C_2 shown above and assume that the students with GPA 3.56, 3.57, 3.57, 3.58, and 3.59 are deleted in that order. C_2 will become C_2: GPA >= (3.59), and the value 3.59 will be flagged as invalid. C_2 now represents an inexact lower bound. Now assume that a student with GPA 3.55 is inserted. The value 3.59 will be replaced by 3.55 and C_2 will represent the lowest GPA in the view, C_2: GPA >= (3.55).

Consider constraints C_3 and C_4 shown below. Each of these constraints involve attributes that require boundary value lists.

C_3: (GPA $\geq$ 3.54) AND (GPA $\leq$ 3.58) $\Rightarrow$ Advisor = Jones (3)

C_4: (SALARY $\geq$ 12K) AND (SALARY $\leq$ 15K) $\Rightarrow$ (GPA $\geq$ 3.2) AND (GPA $\leq$ 3.8)

The counter (3) associated with C_3's condition (Advisor = Jones) indicates that three students satisfy the constraint. Without boundary value lists for GPA, as shown below, the system cannot determine the middle GPA without materializing the view. Constraint C_4 would require four boundary value lists, one list for each term.

C_3: (GPA $\geq$ (3.54, 3.55)) AND (GPA $\leq$ (3.55, 3.58)) $\Rightarrow$ Advisor = Jones (3)

3.2 Refreshing Constraints

This section presents an algorithm for applying an update to an instance-based constraint in order to refresh the constraint. Consider constraint C_5, shown below, which asserts that if a student's GPA is greater than 3.9, then the student is advised by Jones.

C_5: GPA > 3.9 $\Rightarrow$ ADVISOR = JONES

The left hand side of a constraint is a list of terms. Each term represents a condition. The right hand side of a constraint is also a list a terms, but the terms on the right hand side represent assertions. Constraint C_5 has one condition on the left hand side (GPA > 3.9) and one assertion on the right hand side (ADVISOR = JONES).

An update applies to a constraint if it satisfies all of the conditions on the left hand side. As soon as one condition is not satisfied, the update can be disregarded with respect to the constraint being refreshed. If an update satisfies the conditions of the constraint, then the assertions on the right hand side must be considered. In this example, if the update's GPA

is greater than 3.9 and the advisor is Jones then the update does not affect the constraint. However, if the update satisfied the condition (GPA $\geq$ 3.9) and the advisor is not Jones, then there are three options: (1) modify the assertion, (2) modify the condition, or (3) modify both the assertion and the condition by creating another constraint.

For example, assume a student is added to the database with a GPA of 3.92 and the student is advised by SMITH. The student's GPA satisfies the condition of the constraint, but the assertion no longer holds. The following two constraints can be created.

GPA $>$ 3.92 $\Rightarrow$ ADVISOR = JONES

GPA $>$ 3.9 $\Rightarrow$ (ADVISOR = JONES) OR (ADVISOR = SMITH)

The following algorithm applies one update to an instance-based constraint.

```
1   If the update satisfies the condition then
2       If the assertion no longer holds then
3           Modify the constraint
```

As described above, an update may alter the representation of a constraint without affecting the validity of the constraint. For example, a deletion may add or remove a value from the boundary value list. In this case, the constraint would have to be modified, but the assertion would still hold.

Range Constraints

Specific algorithms can be developed to efficiently process some of the constraint types. For example, if there are range constraints to be updated, instead of processing each update individually for each range constraint, a single pass through the updates can collect the high and low values for each attribute with a range constraint. The high and low values can then be applied to each constraint. Furthermore, the pass that collects the high and low values can be performed during the filtering process. As the new updates are filtered, all of the range constraints at a node can be updated with only a few additional operations.

For example, consider a view node that represents graduate students and assume that the lowest GPA in the view is 3.1. Constraint C_6 represents the lower bound on GPA at the view, C_6: GPA $\geq$ 3.8. Assume the following six insertions are received by the system: (grad, 3.3), (grad, 3.0), (frosh, 3.6), (soph, 3.4), (senior, 2.7), (grad, 3.8). In order to update the graduate student view, these updates must be filtered to remove the students that are not graduates. When the updates are filtered (i.e., tested for CLASS = GRAD), the system can maintain the lowest GPA for the graduate student updates. When these six updates are filtered, the three irrelevant updates will be removed and the lowest graduate GPA for the insertions will be recored as 3.0. The lowest GPA for the updates (3.0) will be compared with the GPA range constraint at the view node (3.1), and the range constraint will be modified if the insertions have a lower GPA than the current range constraint. In this example, the range constraint will be modified to reflect the 3.0 GPA that has been inserted.

4 Conclusion

Instance-based constraints are more useful to semantic query optimization because they contain more information than scheme-based constraints. This paper presented a framework for maintaining instance-based constraints. The Metadata View Graph Framework makes three contributions: (1) the framework maintains instance-based constraints for logical views of the database, (2) the framework provides a foundation that directs and integrates existing methods for constraint discovery, and (3) the framework allows instance-based constraints to be retrieved efficiently at run-time.

The problem of maintaining instance-based constraints in the Metadata View Graph can be decomposed into three sub-problems: (1) manage the update logs, (2) manage the view nodes, and (3) refresh the instance-based constraints. This paper analyzed the third sub-problem and considered various representations that improve maintenance efficiency.

Future research will develop efficient strategies for managing update logs and view nodes (i.e., the first two subproblems). Future research will also continue to analyze instance-based constraints in order to develop more efficient update strategies, such as the strategy presented for range constraints, and further classify the properties of instance-based constraints with respect to semantic query optimization and maintenance.

5 Acknowledgements

The author wishes to thank Leo Mark and Shamkant Navathe for their many comments and discussions regarding this research. The author has been generously supported by BNR Inc., the research and development subsidiary of Northern Telecom, and is especially grateful to Robert Bloedon and Deborah Stokes. The author also acknowledges the support of the Advanced Project Research Agency under contract number F33615-93-1-1338. The current work is part of the project entitled: "A Knowledge Based Approach to Integrating and Querying Distributed Heterogeneous Information Systems."

References

[BLT86] J. Blakeley, P. Larson, and F. Tompa. Efficiently updating materialized views. In C. Zaniolo, editor, *Proceedings of the 1986 ACM SIGMOD International Conference on the Management of Data*, pages 61–71, Washington, D.C., May 1986.

[CG94] Richard L. Cole and Goetz Graefe. Optimization of dynamic query evaluation plans. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 150–160, Minneapolis, Minnesota, May 1994.

[CGM90] S. Chakravarthy, J. Grant, and J. Minker. Logic based approach to semantic query optimization. *ACM Transactions on Database Systems*, 15(2):162–207, June 1990.

[Fre87] J.C. Freytag. A rule-based view of query optimization. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 173–180, San Francisco, May 1987.

[GD87] G. Graefe and D. DeWitt. The exodus optimizer generator. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 160–171, San Francisco, May 1987.

[GM93] G. Graefe and W.J. McKenna. The volcano optimizer generator: Extensibility and efficient search. In *Proceedings of the IEEE Conference on Data Engineering*, pages 209–228, Vienna, April 1993.

[GW89] G. Graefe and K. Ward. Dynamic query evaluation plans. In *Proceedings of the 1989 ACM-SIGMOD International Conference on the Management of Data*, pages 358–366, Portland, Oregon, 1989.

[HK93] C. Hsu and C.A. Knoblock. Reformulating query plans for multidatabase systems. In *Proceedings of the Second International Conference on Information and Knowledge Management*, Washington, DC, 1993.

[HK94] Chun-Nan Hsu and Craig Knoblock. Rule induction for semantic query optimization. *Machine Learning*, pages 1–10, 1994.

[HZ80] M. Hammer and S.B. Zdonik. Knowledge-based query processing. In *Proceedings of the Sixth International Conference on Very Large Data Bases*, pages 137–147. Montreal, October 1980.

[Kin81] J. King. Quist: A system for semantic query optimization in relational databases. In *Proceedings of the Seventh International Conference on Very Large Data Bases*, pages 510–517, 1981.

[PMN95] J. Pittges, L. Mark, and S. Navathe. Metadata view graphs: A framework for query optimization and metadata management. *ACM Transactions on Information Systems*, 1995. Unpublished – submitted, Nov. 1994.

[Rou82] N. Roussopoulos. The logical access path scheme of a database. *IEEE Transactions on Software Engineering*, SE-8(6):563–573, November 1982.

[SHKC93] S. Shekhar, B. Hamidzadeh, A. Kohli, and M. Coyle. Learning transformation rules for semantic query optimization: A data-driven approach. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):950–964, 1993.

[SO87] S.T. Shenoy and Z.M. Ozsoyoglu. A system for semantic query optimization. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 16(3):181–195, December 1987.

[SO89] S.T. Shenoy and Z.M. Ozsoyoglu. Design and implementation of a semantic query optimizer. *IEEE Transactions on Knowledge and Data Engineering*, 1(3):344–361, 1989.

[SSD92] Shashi Shekhar, Jaideep Srivastava, and Soumitra Dutta. A formal model of trade-off between optimization and execution costs in semantic query optimization. *Data and Knowledge Engineering*, 8:131–151, 1992.

[SSS92] M. Siegel, E. Sciore, and S. Salveter. A method for automatic rule derivation to support semantic query optimization. *ACM Transactions on Database Systems*, 17(4):563–600, December 1992.

[YS89] C. Yu and W. Sun. Automatic knowledge acquisition and maintenance for semantic query optimization. *IEEE Transactions on Knowledge and Data Engineering*, 1(3):362–375, September 1989.

Maintaining Semantic and Structural Metadata in the Metadata View Graph Framework

Jeff Pittges
Leo Mark
Shamkant B. Navathe

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
{pittges, leomark, sham}@cc.gatech.edu

May 1, 1997

Abstract

The Metadata View Graph is a metadatabase capable of maintaining semantic and structural metadata for views of a database. Semantic metadata provides dynamic rules which are used during query optimization and structural metadata provides indexes which are used during query execution. Since both types of metadata represent the current contents of the database, both types of metadata must be maintained when the contents of the database change.

Although both types of metadata use the same update logs, these logs are typically processed twice because the semantic metadata must be maintained *before* query execution while structural metadata is usually maintained *during* query execution. However, when a query execution plan requires both types of metadata, it is most efficient to process the update logs once and maintain both types of metadata at the same time. This creates a conflict when the update paths for the semantic and structural metadata overlap.

This paper presents several methods for efficiently maintaining semantic and structural metadata. The paper analyzes the maintenance cost for both types of metadata and proposes two strategies for processing overlapping update paths.

Keywords: Metadata Maintenance, View Maintenance, Constraint Maintenance, Dynamic Semantic Rules, View Cache, Query Optimization.

1 Introduction

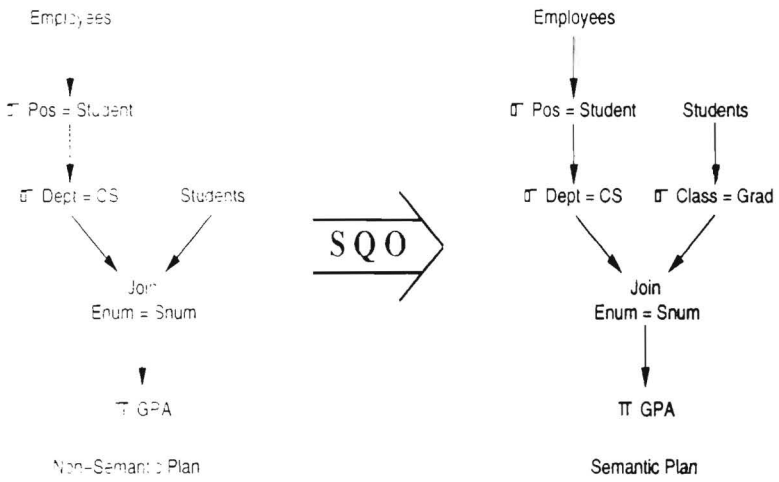
Dynamic rules (semantic metadata) are used by semantic query optimization [HK94, HK93, S+93, S+92, CGM90, SO89, YS89] to reformulate a query into a semantically equivalent query that is more efficient to execute. Since dynamic rules represent the current contents of the database, dynamic rules must be maintained before the query is executed in order to guarantee that the reformulated query is correct. For example, consider a query that requests the grade point average of the students who work in the computer science department. Figure 1 illustrates two query execution plans for this query. The *non-semantic plan* is produced directly from the original query expression. The *semantic plan* is produced by applying transformation rule R_{35} which states that only graduate students work in the computer science department.

The non-semantic plan performs the join operation with the entire Student base relation whereas the semantic plan only joins the graduate students. More importantly, the semantic plan may greatly reduce the number of pages retrieved from the Student base relation if the database maintains an index on the CLASS attribute. Note, however, that the dynamic rule (R_{35}) is only valid for particular instances of the database. If an undergraduate student from the CS department is later employed by the school, then the rule is no longer valid and the semantic plan cannot be used to answer the query. Therefore, the dynamic rule must be maintained before the query is executed.

Incremental query computation [Rou91] maintains views in order to answer queries more efficiently. An index (structural metadata) called a *view cache* maintains a set of pointers into the base relation tuples that belong to a given view. The view cache is used to materialize the view efficiently during query execution. For example, consider a view representing a join between two base relations, R_i and R_j. For each tuple in the view, the view cache will contain a pair of tuple IDs $\langle TID_i, TID_j \rangle$ where TID_i and TID_j are pointers to the tuples in R_i and R_j that form the tuple in the view. View caches are maintained incrementally during query execution by propagating relevant updates while the view is being materialized. Updating the view cache is often more efficient than recomputing

Relation	Attributes
Students	Snum, Class, GPA, Advisor
Employees	Enum, Salary, Dept, Pos

Q_1: Select GPA
From Employees, Students
Where Pos = Student AND Dept = CS
AND Enum = Snum



R_{35}: (Pos = Student) AND (Dept = CS) $\Rightarrow$ Class = Grad

Figure 1: Reformulating a query with semantic query optimization.

a view. For example, consider query Q_1 above and assume the system maintains a view cache representing the graduate students in the CS department who are employed. The view cache could be used to answer the query by materializing the view and projecting the grade point averages. Note, however, that this view cache can only be used if rule R_{35} is valid.

Since dynamic rules are used to reformulate the original query, the rules must be maintained *before* the query is executed. Since view caches are more efficient to maintain when the views are materialized, view caches should be maintained *during* query execution. However, dynamic rules and view caches use the same update logs. Therefore, when a query requires both types of metadata, it is most efficient to process the update logs once and maintain the dynamic rules and view caches at the same time. This paper presents several methods for efficiently maintaining semantic and structural metadata.

The rest of the paper is organized as follows. Section 2 presents the Metadata View Graph Framework and describes how metadata is stored, retrieved, maintained, and used during query processing. The framework is presented to help the reader appreciate the problems addressed by the paper. Section 3 specifies the maintenance problem, analyzes the maintenance cost for both types of metadata, and proposes several strategies and algorithms for maintaining semantic and structural metadata. The last section contains our concluding remarks.

2 The Metadata View Graph Framework

As illustrated in Figure 2, the Metadata View Graph [Pit95] is a metadatabase capable of maintaining semantic, statistical, and structural metadata for views of a database. The Metadata View Graph consists of four components: (1) a lexicon, (2) a semantic network, (3) a view network, and (4) a QEP Network. The lexicon maintains the system's vocabulary and contains information for each word or phrase that is recognized by the system interface. The semantic network captures domain knowledge and can be used to disambiguate queries. The lexicon and semantic network are used for query interpretation and will not be discussed further.

The View Network is an extension of Roussopoulos' Logical Access Path schema [Rou82]. The view nodes in the network ($V_1 - V_{11}$ in Figure 2) represent views of the database and store metadata specific to the particular data set. The links represent logical operations and semantic relationships. The View Network is essentially a collection of query graphs overlaid on top of each other with each view node representing an intermediate result.

The QEP Network maintains a separate *query execution packet* for each application query. Query execution packets store two types of query execution plans, ***non-semantic plans*** and ***semantic plans.*** Non-semantic plans are produced by conventional query optimization and do not require maintenance. Semantic plans are produced by semantic query optimization. Consequently, semantic plans depend on the dynamic rules that were used to reformulate the original query. These rules must be

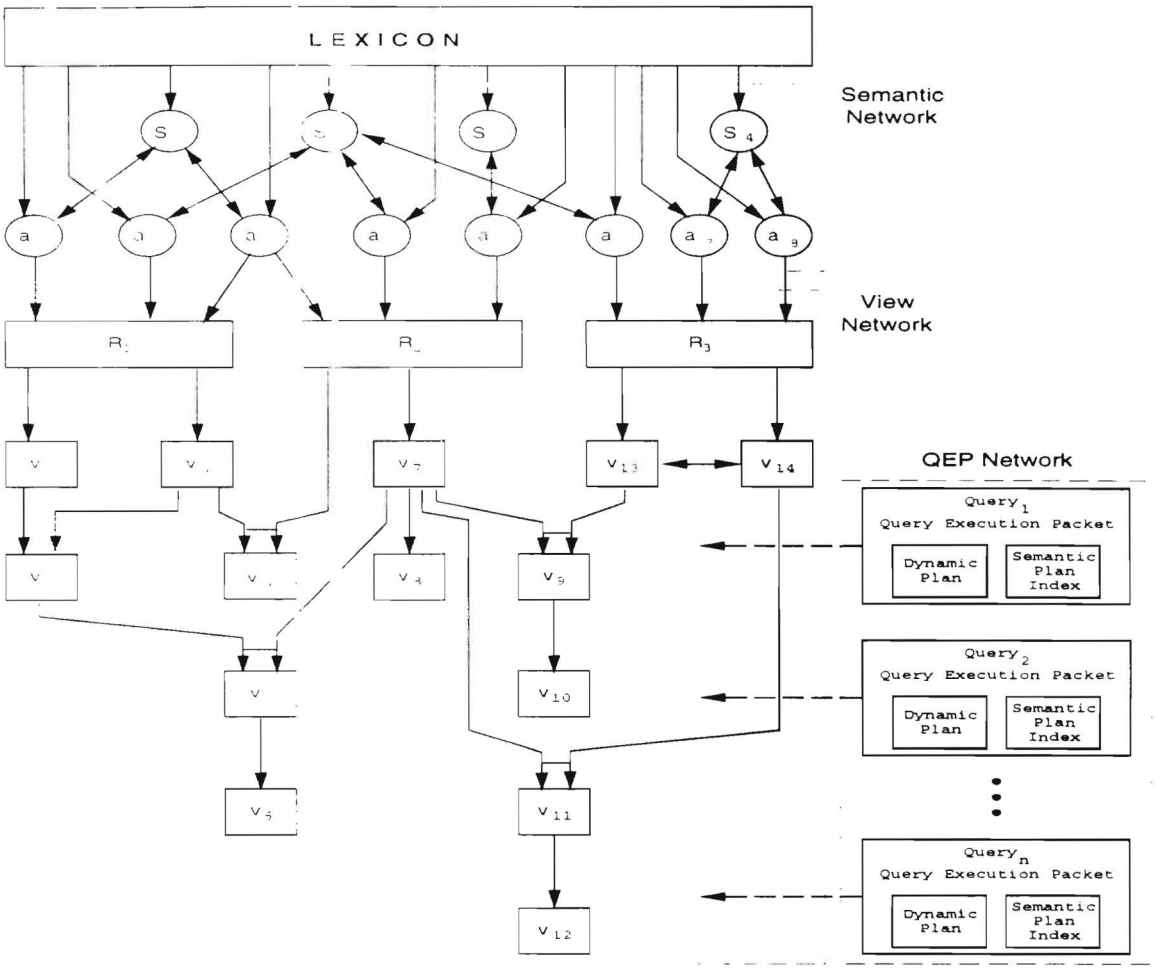


Figure 2: Conceptual representation of the Metadata View Graph.

updated and validated before a semantic plan can be executed. Therefore, semantic plans contain pointers to the rules that they depend on in order to efficiently retrieve and maintain those rules before the query is executed.

2.1 Query Optimization

When a query is processed at compile-time, several query execution plans are generated and metadata is collected for each query execution plan. We assume that existing methods have been used to discover the dynamic rules that are used to generate semantic plans. The most efficient query

executable plans are stored in the QEP Network. When a query is processed at run-time, the query's executable packet is retrieved from the QEP Network along with any relevant view nodes from the View Network. The semantic plans are ordered such that the run-time bindings of the query can be used to select the semantic plans that match the conditions of the query.

An Example

Figure 3 illustrates part of a View Network that supports our example query Q_1. The figure also shows semantic plan SP_4 which depends on rule R_{35} stored at node V_3. Rule R_{35} was used to recommute Q_1 by introducing the selection condition CLASS = GRAD. When Q_1 is processed at run-time, semantic plan SP_4 will be retrieved from the QEP Network along with view node V_3. If maintenance is cost effective, the metadata at node V_3 will be maintained, and if rule R_{35} is valid, then SP_4 will be executed. If R_{35} is not valid, the non-semantic plan for Q_1 will be executed.

Relation	Attributes
Students	snum, class, GPA, advisor
Employees	enum, salary, dept, pos

Q_1: Select GPA
From Employees, Students
Where pos = student AND dept = var_1 AND enum = snum

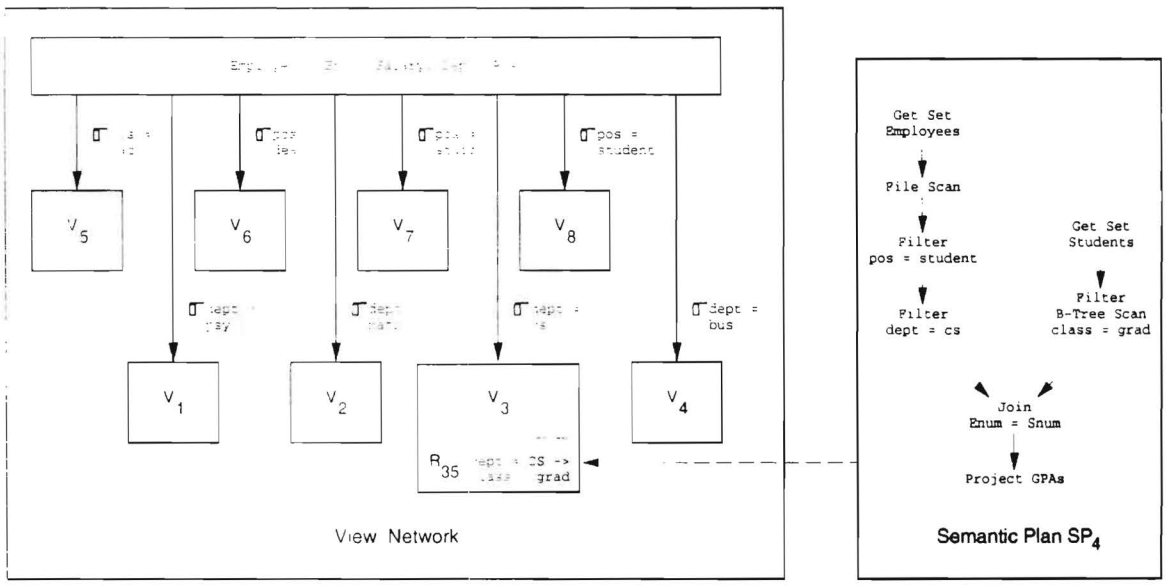


Figure 3: Semantic plan SP_4 depends on dynamic rule R_{35} which is stored in the View Network.

3 Maintaining Semantic and Structural Metadata

When a database evolves from one database state x to another database state y due to an update (i.e., a tuple is inserted, deleted, or modified), the metadata in the View Network may become invalid. In general, two actions must occur before the metadata can be used: (1) the metadata that is affected by the update must be identified, and (2) if the metadata is no longer valid, the metadata must be modified or invalidated.

Rather than consider each update as it is received, it is more efficient to process a batch of updates for a particular set of metadata when the metadata is needed. This is referred to as the *deferred update strategy* [Rou91]. This approach requires the system to maintain an update log for each base relation. When an update is made to a base relation, the system writes the update to the base relation's update log. When the metadata at a view node needs to be maintained, all of the updates for the base relations from which the view is derived are processed. However, only a subset of the updates will apply to a given view node based on the definition of the node (e.g., GPA $\geq$ 3.0). Therefore, the updates must be *filtered* to remove the *irrelevant updates* [BLT86].

Figure 1 illustrates the contents of a base relation and a view node for two states of an example database. The figure also illustrates an update log for the base relation which contains a number of tuples to be inserted into the base relation. State x represents the database before the updates are received, and state y represents the database after the updates have been processed and the metadata at the view node has been maintained.

The view node shown in this example contains seven rules, a tuple count, a distribution profile, and a view cache pointer (since the pointer is nil, there is no view cache for this node). The view is defined for students with a GPA of 3.0 or greater. The rules and tuple count at the view node can be verified for states x and y by selecting the tuples from the base relation (as shown) that satisfy the definition of the node (i.e., GPA $\geq$ 3.0). An update contains a unique time-stamp, which indicates when the update was received, along with the tuple to be inserted, deleted, or modified.

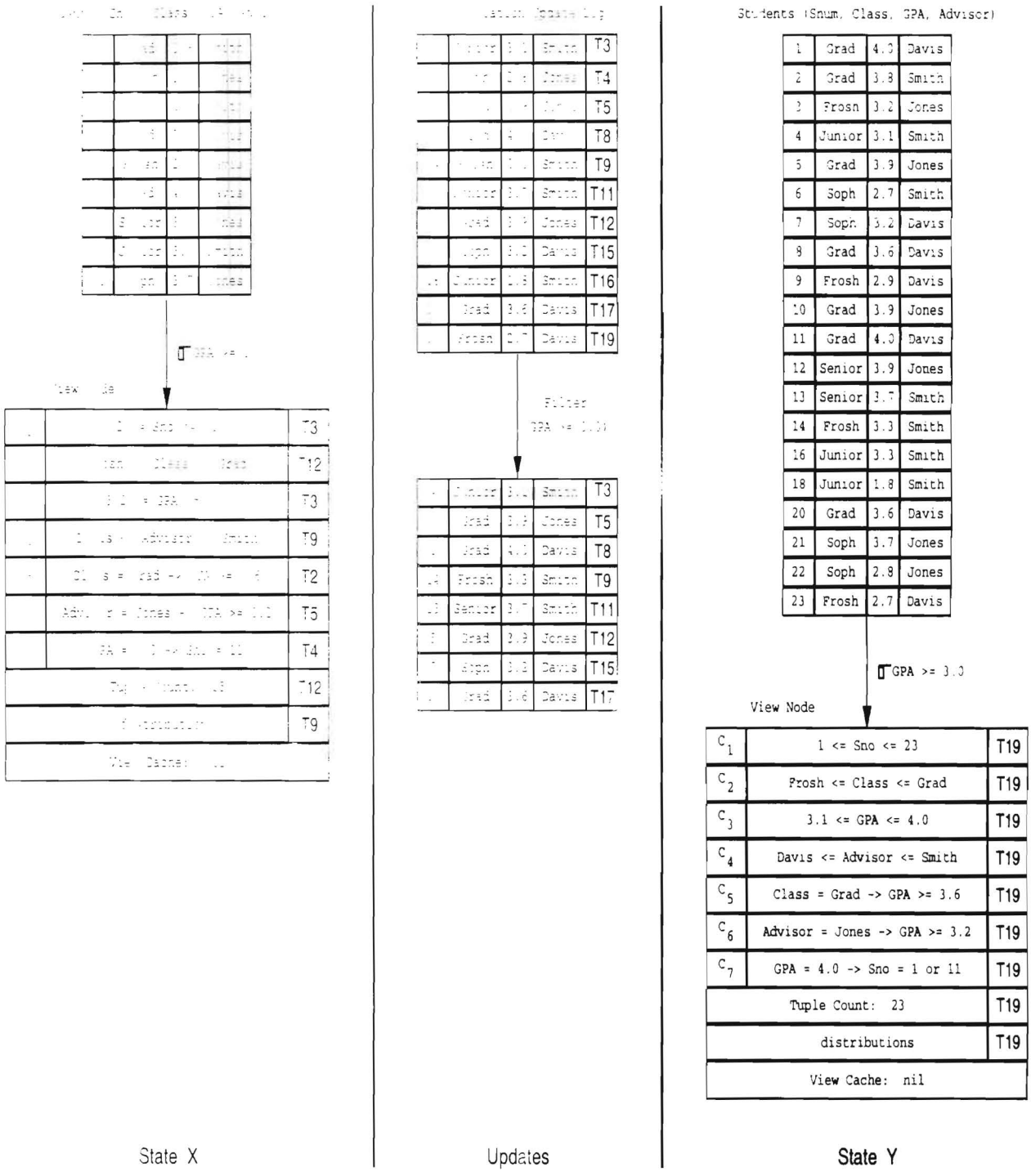


Figure 4: The base relation update log contains a set of insertions that are filtered and applied to the view node as the database evolves from state x to state y.

By propagating the updates through the View Network, the system can identify which metadata is affected by an update. When an update arrives at a node, the metadata can be tested to determine if the metadata is no longer valid, and the system can modify or invalidate the incorrect metadata. Before the updates can be filtered and propagated through the View Network, the update logs and the view nodes to be maintained must be retrieved from disk. The cost of these disk accesses dominates the cost of the maintenance process.

Allocating and Processing Update Logs

Our objective is to minimize the average maintenance cost per query. This is achieved by creating additional update logs in the View Network. The additional logs form a chain in which each log only considers the updates in the log above. Once an update is filtered, it is never considered by an update log lower in the network. Although this strategy may create additional work for an individual query, the maintenance cost is amortized thus reducing the average cost per query.

Update logs are allocated throughout the View Network at compile-time based on the estimated selectivity of the view nodes. As described in [Pit95], when the selectivity at a node, with respect to the log above, is less than 50 percent (i.e., less than half of the updates apply to the node), an update log is created at the view node. These logs store relevant updates in order to reduce the number of log pages that must be read to maintain the rest of the network during future maintenance cycles.

The cost of maintaining the view nodes is dominated by the cost of reading and writing the update logs. Log maintenance is also a significant factor in the cost of maintaining view caches. Although the view nodes and view caches are separate data structures, they both depend on the same update logs. Therefore, maintenance costs can be minimized by efficiently utilizing the update logs (i.e., maintaining the view nodes and view caches together when an update log is retrieved).

Running Example

Before we present the cost formulas for maintaining view nodes and view caches, the following example provides some numbers to help interpret the formulas. This example considers the chain of nodes V_1, V_2, V_3 and V_4 shown in Figure 5. We will continue to denote the nodes in the View Network as V unless it is necessary to distinguish between the view nodes (VN_i) and view caches (VC_i). As shown in Table 1, we assume that the base relation contains one million tuples. We estimate the tuple count for each of the four views by assuming that the selectivity factor at each level of the network is 50 percent (i.e., exactly half of the tuples at a given node apply to the nodes directly below).

	Tuples	Log Tuples	Log Pages	View Cache Pages (n_v)	View Cache Partitions (p_v)
Base Relation	1,000,000	10,000	1000		
Node V_1	500,000	5000	500		
Node V_2	250,000	2500	250	1000	50,000
Node V_3	125,000	1250	125		
Node V_4	62,500			250	6250

Table 1: Parameters for the running example.

In order to estimate the number of log pages that will be read and written, we assume that one percent of the base relation has changed. Therefore, the base relation update log contains 10,000 updates. Assuming 10 updates per page, 1000 pages will be read from the base relation update log. The number of updates for each view were estimated assuming a 50 percent selectivity factor between each view. In order to estimate the number of pages in the view caches (n_v), we assume that each pointer requires 5 bytes and that each TID in the view cache requires 3 pointers. Therefore, each entry in the view cache requires 15 bytes. Assuming a page size of 4K, each view cache page contains 250 entries. In order to estimate the number of partitions (p_v) in the view cache, we assume that VC_2 references half of the base relation pages (50,000) and we assume that each tuple in VC_4 is written on its own page (6250 partitions). Finally, we assume that each view node requires one page ($N_v = 1$).

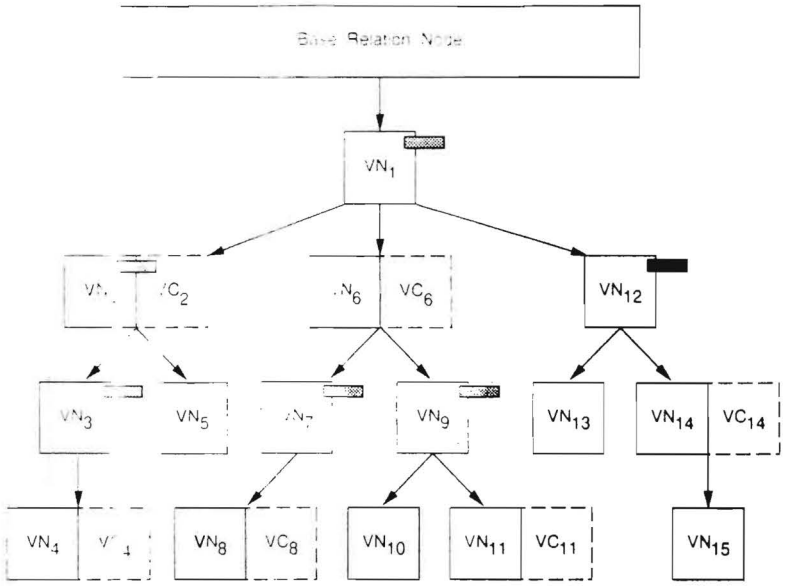


Figure 5: Example View Network with view nodes, view caches, and update logs.

Maintaining View Nodes

In order to maintain a view node, all of the logs along the update path must be retrieved and processed. Processing begins at the top of the network with the base relation update log. The updates in the base relation update log are filtered and the relevant updates are written to the next log. This process continues until the last update log in the update path has been processed. The updates in the last log that have not been applied to the metadata at the view node being maintained are applied to the metadata. The following formula estimates the cost of maintaining a view node.

$$L_r + L_w + 2N_r$$

L_r represents the number of log pages read. L_w represents the number of log pages written, and N_r represents the number of node pages that are read. In the worst case, each node page will be modified and written back to disk. The cost formula allows for this by doubling the number of node pages that are read. In our running example, the cost of maintaining VN_4 is $L_r + L_w + 2N_r = 1000 + 875 + 2(1) = 1877$.

From these numbers, it is clear to see that the cost of updating a view node is dominated by the cost of reading and writing the update logs (1875 pages to process the update logs versus 2 pages to read and write the view node). Therefore, whenever the update logs are processed, all of the view nodes along the update path should be maintained. The view nodes along the update path for V_1 (V_1 ⋈ V_2, and V_1 ⋈ V_2 ⋈ ...) can be maintained by reading (and possibly writing) 3 more pages.

Maintaining View Caches

View caches contain pointers to the base relation tuples that participate in a view. When an update is made that affects the view, the view cache must be maintained. (i.e., the insertions that apply to the view must be added to the view cache and the deletions must be removed). Maintaining a view cache is typically more efficient if the view is materialized, especially if the view joins two or more base relations.

With the deferred update strategy, view caches are updated when the view is materialized during query execution. Three items are retrieved when a view cache is maintained: (1) the view cache, (2) the update logs along the update path that support the view cache, and (3) the pages of the base relation that are indexed by the view cache. A view cache is *optimal* if it can be materialized without reading the same base relation page more than the minimum number of times required. This is achieved by partitioning the TIDs of the view cache into equivalence classes in which all of the TIDs access the same base relation page.

[Rou91] provides a detailed analysis of incremental update algorithms for view caches. The cost of these algorithms depends on four parameters: (n_v) the number of disk pages in the view cache V being maintained (L_r) the number of log pages read, (L_w) the number of log pages written, and (p_v) the number of partitions in V. The following formula estimates the cost of materializing a unary view cache (selection or projection).

$$L_r + L_w + 2n_v + p_v$$

Each view cache page may be modified and written back to disk. The cost formula allows for this by doubling the number of view cache pages that are read. Since the view cache is updated during query execution, the cost of materializing the view ($n_v \cdot p_r$) is absorbed by the cost of executing the query. Therefore, the actual maintenance cost consists of reading and writing the update logs plus the cost of reading the modified pages of the view cache ($L_r + L_w + n_v$). In our running example, the cost of maintaining VC_{14} is $L_r + L_w + 2n_v \cdot p_r = 1000 + 875 + 2(250) + 6250 = 8625$.

These numbers indicate that the cost of updating a view cache is dominated by the cost of retrieving the tuples from the base relations (p_r). However, in this example we have considered the worst case (i.e., that each tuple is stored on its own page). If we assume that each partition contains two tuples, the cost of retrieving the tuples is cut in half ($p_v = 3125$), and the cost of reading and writing the update logs ($L_r + L_w$) begins to approach the cost of materializing the view. As the size of the views decreases, the cost of reading and writing the update logs becomes a greater factor.

3.1 Maintaining View Nodes and View Caches

Having reviewed the methods for maintaining view nodes and view caches independently, the remainder of this paper considers how to efficiently maintain them together. Since the cost of maintaining view nodes and view caches greatly depends on the cost of reading and writing the update logs, it is desirable to maintain the metadata at the view nodes and view caches whenever an update log is retrieved. For example, in Figure 5, if a query requires the metadata at VN_{14} to be maintained, then the base relation update log, plus the update logs at VN_1 and VN_{12}, must be retrieved. When these update logs are processed, it would be most efficient to maintain VN_1, VN_{12}, VN_{13}, VC_{14}, and VN_{15}. However, since the query only requires the metadata at VN_{14}, the additional view nodes and view caches can be maintained if time permits.

The following analysis assumes that a query has been received at run-time and the query's execution package has been retrieved. Within this context, this section will consider three maintenance

scenarios. In all three scenarios, the query execution plan under consideration uses a view cache. In the first scenario, the query execution plan does not depend on any rules. Therefore, the view cache will be maintained during query execution and the view nodes along the update path may be maintained when the update logs are processed. In the second and third scenarios, the query execution plan does depend on rules which must be updated before the query execution plan can be executed. In the second scenario, the update paths for the view node and the view cache do not overlap. Therefore, each update path can be maintained separately. In the third scenario, the update paths do overlap which creates an interesting conflict.

Scenario 1: Non-Semantic Plans

In the first scenario, the query optimizer has selected a non-semantic plan that uses a view cache. Since the plan does not depend on any rules, none of the view nodes need to be maintained. Therefore, the view cache is maintained during query execution as described above. When the update logs are retrieved and processed to maintain the view cache, there is an opportunity to maintain the view nodes along the view cache update path. The cost of maintaining the view cache is: $L_r - L_w + 2n_v - p_v$.

Since the update logs have already been retrieved, the cost to maintain the view nodes is the cost of reading and writing the view node pages ($2N_r$ in the worst case). The total cost to update the semantic and structural metadata along the view cache update path is: $L_r + L_w + 2n_v + p_v + 2N_r$. Since the cost of maintaining the view nodes is dominated by the cost of reading and writing the update logs, the view nodes along the view cache update path should always be maintained whenever the view cache is maintained.

Figure 6 illustrates a View Network and two query execution plans. The non-semantic plan uses the view cache VC_2 shown in bold. Since this plan does not depend on any rules, VC_2 will be maintained during query execution. When the update logs are processed for VC_2, the view nodes along the update path (VN_1 and VN_2) can be maintained. Since the logs have already been

Select Snum
From Students, Employees
Where Advisor = Jones AND Class = Grad AND Snum = Enum

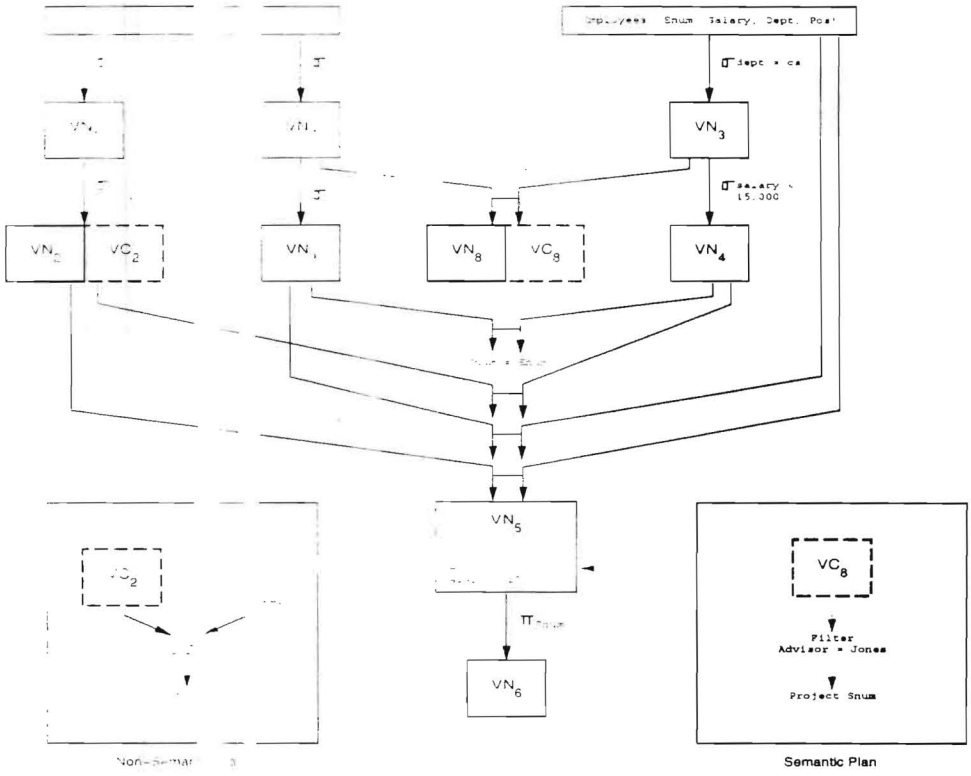


Figure 6: Example View Network and Query Execution Plans.

retrieved and processed, the cost of maintaining the metadata at VN_1 and VN_2 is the cost of reading and writing VN_1 and VN_2.

There are two advantages to maintaining the view nodes when the view caches are maintained. First, although there is no benefit to the current query in this scenario, maintaining the view nodes will reduce the maintenance cost for future queries. Second, rules can be recomputed if the view is materialized when the view node is maintained. For example, consider a rule that maintains the minimum GPA for a view of students and assume that the student with the lowest GPA is deleted. If the view is materialized when the rule is maintained, the new minimum GPA can be recomputed.

```
INPUT:  LOGS,          /* a list of update logs to be processed */
        NODES,         /* a list of node groups to be maintained */
        VIEW-CACHES:   /* a list of view caches to be maintained */

view-cache-update (LOGS, NODES, VIEW-CACHES)
    For each log in LOGS
        read the log
        filter the log
        If there are updates to be added to the log
            write the log
        maintain the log's node group
        If the log server has a view cache in VIEW-CACHES
            maintain the view cache
```

Algorithm 1: View Cache Update.

Scenario 2: Distinct Update Paths

The second scenario involves a semantic plan that depends on one or more rules stored at a view node. Therefore, the view node must be maintained before the semantic plan can be executed. In this scenario the update paths for the view node and the view cache are distinct. Consequently, the update path for the view node can be processed before the query is executed and the update path for the view cache can be processed during query execution if the semantic plan is valid.

In Figure 6, the semantic plan uses the view cache VC_8 which contains an index for the graduate students who are employed by the computer science department. The semantic plan is only valid if the rule stored at VN_5 is valid. There are two possible update paths for VN_5, one of which will be chosen at compile-time. If the update path for VN_5 goes through VN_1 and VN_2, then the update path for VN_5 does not overlap with the update path for VC_8. This scenario involves two steps: (1) maintain the view node, and (2) maintain the view cache provided the semantic plan is valid.

Step 1 (Maintain the View Node): In this example, the metadata at VN_5 will be maintained before the query is executed. View nodes VN_1, VN_2, and the view cache VC_2 can be maintained when the update logs are retrieved for VN_5. The total cost to maintain the view nodes and view

```
INPUT: LOGS,        /* a list of update logs to be processed */
       NODES,       /* a list of node groups along the update path */
       VIEW-CACHES: /* a list of view caches along the update path */

view-node-update (LOGS, NODES, VIEW-CACHES)
  for each log in LOGS
    read the log
    filter the log
    if there are updates to be added to the log
      write the log
    if time permits
      maintain the log's node group
    if the log serves a view cache in VIEW-CACHES AND time permits
      maintain the view cache
```

Algorithm 2: View Node Update.

caches is: $L_r + L_w - 2N_r + 2(n_r + p_r)$, where $(2n_r + p_r)$ is the additional cost for the view caches.

Step 2 (Maintain the View Cache): The view cache VC_8 will be maintained during query execution. The view nodes VN_7 and VN_9 can be maintained when the update logs are retrieved to maintain VC_8. The total cost to maintain the view nodes and view caches is: $L_r + L_w + (2N_r) + 2(n_r + p_r)$ where $(2N_r)$ is the additional cost to maintain the view nodes.

Scenario 3: Overlapping Update Paths

The third scenario is identical to the second scenario except that the update paths overlap (i.e., the update paths for the view node and view cache contain a common subpath). When the common subpath is processed, it is most efficient to retrieve the update logs once and process the view nodes and view caches together. However, since the plan being considered is a semantic plan, the view cache being maintained may not be used if the semantic plan is invalid.

In Figure 6, if the update path for VN_5 goes through VN_7 and VN_9, then the update path for VN_5 overlaps with the update path for VC_8. In this case, the update logs for VN_7 must be retrieved before the query is executed. Although it is most efficient to maintain VC_8 when the update log

```
INPUT: LOGS,          /* a list of update logs to be processed */
       NODES,         /* a list of node groups along the update path */
       VIEW-CACHES:   /* a list of view caches along the update path */

view-node-view-cache-update (LOGS, NODES, VIEW-CACHES)
  For each log in LOGS
    read the log
    lock the log
    If there are updates to be added to the log
      write the log
    maintain the log's node group
    If the log server is *
      maintain the view cache
    Else
      If the log server is a view cache in VIEW-CACHES AND time permits
        maintain the view cache
```

Algorithm: Combined view node and view cache update.

For VN_7's processed, the view cache VC's should be maintained during query execution. If the rule at VN_5 is invalid, VC's will not be used to answer the query.

There are two solutions for this scenario, an optimistic approach and a pessimistic approach. The optimistic approach assumes that the semantic plan will be valid (i.e., the rule holds for the current state of the database). Therefore, VC's will be materialized and maintained when the update logs are processed for VN_7. If necessary, the materialized view will then be stored on disk while the metadata at VN_5 is maintained. If the rule at VN_5 holds, then the semantic plan will be executed using the view cache. Otherwise, the non-semantic plan will be executed. The optimistic approach maintains the semantic and structural metadata before the query execution plan is selected. If the semantic plan is invalid, then the optimistic approach pays the price of maintaining a view cache that cannot be used for the given query. Although this will reduce the maintenance cost for future queries that use VC's, this can be an expensive price to pay for the current query.

The pessimistic approach assumes that the semantic plan will not be valid. Therefore, the metadata for VN_7 is maintained before a query execution plan is selected. If the semantic plan is valid, then the low level caches at VC_5 will be maintained. This may require the update log for VN_7 to be retrieved a second time, but since the update log has already been processed, only the updates that are relevant to VN_7 will be retrieved. The cost of retrieving the update log is only significant if the size of the log approaches the size of the view cache.

4 Conclusion

Previous researchers have developed methods for discovering dynamic rules and for providing useful view caches. This paper has considered the problem of maintaining these two types of metadata together in the Metadata View Graph framework which can be implemented as a database application or as an extension to the DBMS kernel. Because these two types of metadata are maintained at different times, a conflict arises when the update paths for the semantic and structural metadata overlap. Since both types of metadata depend on the same update logs, it is most efficient to maintain the metadata together when the update logs are retrieved. This paper presented various cost formulas and analyzed three scenarios. The first scenario showed that the view nodes along a view cache update path should always be maintained whenever the view cache is maintained. The second scenario showed that it is only cost effective to maintain a view cache along a view node update path when the size of the logs approaches the size of the view cache.

The third scenario presents a conflict because the update paths overlap. We have proposed two approaches to resolve this conflict. The optimistic approach is the most efficient, but the current query may incur a heavy penalty when the semantic plan is invalid. The pessimistic approach may be less efficient when the semantic plan is valid, but the additional cost is minimized by the previous maintenance cycle. The best approach can be selected at run-time based on cost estimates and other factors.

References

[BLT86] J. Blakeley, P. Larson, and F. Tompa. Efficiently updating materialized views. In C. Zaniolo, editor, *Proc. of the 1986 ACM SIGMOD*, pages 61–71, Washington, D.C., May 1986.

[CGM90] S. Chakravarthy, J. Grant, and J. Minker. Logic based approach to semantic query optimization. *ACM Trans. on Database Systems*, 15(2):162–207, June 1990.

[HK93] C. Hsu and C. A. Knoblock. Reformulating query plans for multidatabase systems. In *Proc. of the Second Int. Conf. on Info. and Know. Management*, Washington, DC, 1993.

[HK94] Chun-Nan Hsu and Craig Knoblock. Rule induction for semantic query optimization. *Machine Learning*, pages 1–10, 1994.

[Pit95] J. Pittges. *Metadata View Graphs: A Framework for Query Optimization and Metadata Management*. PhD thesis, Georgia Institute of Technology, November 1995.

[Rou82] N. Roussopoulos. The logical access path scheme of a database. *IEEE Trans. on Software Eng.*, SE-8(6):563–573, November 1982.

[Rou91] N. Roussopoulos. An incremental access method for viewcache: Concept, algorithms, and cost analysis. *ACM Trans. on Database Systems*, 16(3):535–563, 1991.

[S+92] M. Siegel et al. A method for automatic rule derivation to support semantic query optimization. *ACM Trans. on Database Systems*, 17(4):563–600, December 1992.

[S+93] S. Shekhar et al. Learning transformation rules for semantic query optimization: A data-driven approach. *IEEE Trans. on Knowledge and Data Eng.*, 5(6):950–964, 1993.

[SO89] S. T. Shenoy and Z. M. Ozsoyoglu. Design and implementation of a semantic query optimizer. *IEEE Trans. on Knowledge and Data Eng.*, 1(3):344–361, 1989.

[YS89] C. Yu and W. Sun. Automatic knowledge acquisition and maintenance for semantic query optimization. *IEEE Trans. on Knowledge and Data Eng.*, 1(3):362–375, September 1989.