EXHIBIT O

08/318252 | 395 | Subclass | ISSUE CLASSIFICATION

5694593

UTILITY SERIAL NUMBER	PATENT DATE	PATENT NUMBER
08/318252	DEC. 02 1997	

SERIAL NUMBER	FILING DATE	CLASS	SUBCLASS	GROUP ART UNIT	EXAMINER
08/318,252	10/05/94	395	600	~~2317~~ 2307	Lewis

APPLICANTS
KENNETH P. BACLAWSKI, WALTHAM, MA.

CONTINUING DATA********
VERIFIED
Examiner's Initials: none

FOREIGN/PCT APPLICATIONS********
VERIFIED
Examiner's Initials: none

FOREIGN FILING LICENSE GRANTED 11/04/94 ***** SMALL ENTITY *****

Foreign priority claimed 35 USC 119 conditions met	Verified and Acknowledged Examiner's Initials	AS FILED	STATE OR COUNTRY	SHEETS DRWGS.	TOTAL CLAIMS	INDEP. CLAIMS	FILING FEE RECEIVED	ATTORNEY'S DOCKET NO.
☐ yes ☑ no ☐ yes ☐ no	CL	→	MA	13	17	6	$479.00	NUB60XX

ADDRESS
WEINGARTEN SCHURGIN GAGNEBIN & HAYES
TEN POST OFFICE SQUARE
BOSTON MA 02109

TITLE
DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

U.S. DEPT. of COMM.-Pat. & TM Office — PTO-436L (Rev. 10-78)

PARTS OF APPLICATION FILED SEPARATELY

Applications Examiner

NOTICE OF ALLOWANCE MAILED	Assistant Examiner: Cheryl R. Lewis	CLAIMS ALLOWED: Total Claims 17 / Print Claim 1
6/19/97		
ISSUE FEE: Amount Due $1605.00 / Date Paid 9/18/97	THOMAS G. BLACK SUPERVISORY PATENT EXAMINER GROUP 2300 Primary Examiner	DRAWING: Sheets Drwg. 13 / Figs. Drwg. 13 / Print Fig. 1
Label Area	PREPARED FOR ISSUE	ISSUE BATCH NUMBER 031

WARNING: The information disclosed herein may be restricted. Unauthorized disclosure may be prohibited by the United States Code Title 35, Sections 122, 181 and 368. Possession outside the U.S. Patent & Trademark Office is restricted to authorized employees and contractors only.

Form PTO-436A (Rev. 10/92)

ISSUE FEE IN FILE GH

(FACE)

JAR0002539

EXHIBIT NO. 17
M. EVANS

08/315252

PATENT APPLICATION

08/315252

CONTENTS

Date Entered or Counted			Date Received or Mailed
	1.	Application ____ papers.	RECEIVED NOV 15 1994 GROUP 2300
	2.	Info. Disc. Stmt.	11-21-94
4/2	3.	Rejection 3mos w/ Att	4-16-96
	4.	Amdt. - A	6-10-96 Ctm 6-7
9-9-96	5.	Rejection 3mos w/ Att	9/11/96
	6.	Amdt B	12-13-96 CM 12-4
3-17-97	7.	FOA 3mos w/ Att	3-21-97 LB
	8.	Examiner's Amendment	5-19-97
6-7-97 =	9.	Notice of allow	6/19/97 LB
9/5/97	10.	Formal Drawings (13 sht(s)) set 1	7/7/97
	11.	PTO Grant DEC 02 1997	
	12.		
	13.		
	14.		
	15.		
	16.		
	17.		
	18.		
	19.		
	20.		
	21.		
	22.		
	23.		
	24.		
	25.		
	26.		
	27.		
	28.		
	29.		
	30.		
	31.		
	32.		

(FRONT)

Staple Issue Slip Here

POSITION		ID NO.	DATE
CLASSIFIER		699	10/25/94
EXAMINER			11-2
TYPIST			11-4-94
VERIFIER			11-5-94
CORPS CORR.			
SPEC. HAND			
FILE MAINT.			
DRAFTING			

INDEX OF CLAIMS

Claim Final	Claim Original	Date
1	1	✓
2	2	-
3	3	✓
4	4	✓
5	5	✓
6	6	✓
7	7	✓
8	8	✓
9	9	✓
10	10	✓
11	11	✓
12	12	✓
13	13	✓
14	14	✓
15	15	✓
16	16	✓
17	17	✓
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	
	50	

SYMBOLS

✓ ..... Rejected
= ..... Allowed
- (through number) Canceled
R ..... Restricted
N ..... Non-elected
I ..... Interference
A ..... Appeal
O ..... Objected

Claim Final	Claim Original	Date
	51	
	52	
	53	
	54	
	55	
	56	
	57	
	58	
	59	
	60	
	61	
	62	
	63	
	64	
	65	
	66	
	67	
	68	
	69	
	70	
	71	
	72	
	73	
	74	
	75	
	76	
	77	
	78	
	79	
	80	
	81	
	82	
	83	
	84	
	85	
	86	
	87	
	88	
	89	
	90	
	91	
	92	
	93	
	94	
	95	
	96	
	97	
	98	
	99	
	100	

(LEFT INSIDE)

[redacted]

SEARCHED

Class	Sub.	Date	Exmr.
395	600	2/14/96	CL
382.1	349.19	4/24/96	CL
348	200	4/24/96	CL
395	600 603 605 621 622	6/11/97	CL

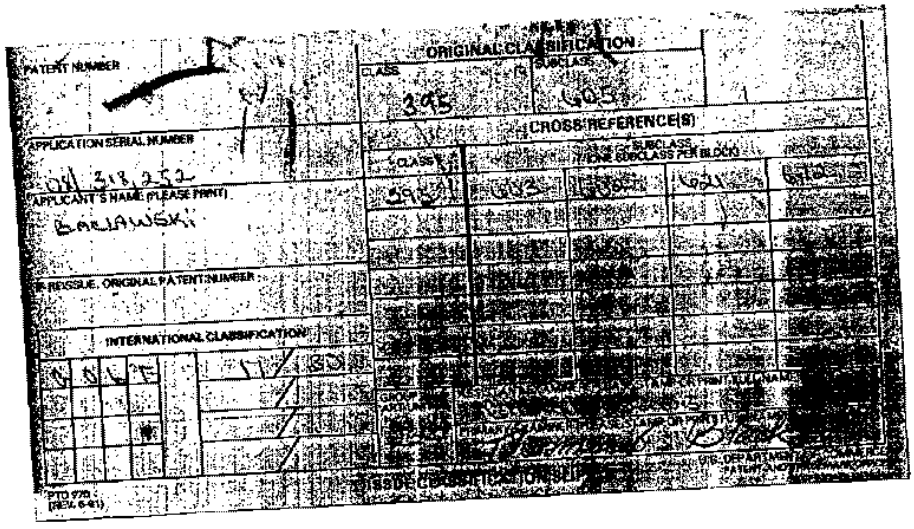
INTERFERENCE SEARCHED

Class	Sub.	Date	Exmr.
395	600 603 605 621 622	6/11/97	CL

SEARCH NOTES

	Date	Exmr.
SEARCHED 600, RESULTS PRINTED IEEE		
updated search	2/14/96	CL
SEARCHED 395 RESULTS PRINTED	4/24/96 5/13/97	CL CL
updated search results enclosed	6/11/97	CL

(RIGHT OUTSIDE)


PATENT NUMBER
ORIGINAL CLASSIFICATION
CLASS
SUBCLASS
395
403
CROSS REFERENCE(S)
APPLICATION SERIAL NUMBER
08/318,252
APPLICANT'S NAME (PLEASE PRINT)
BARJAWSKI
RE-ISSUE, ORIGINAL PATENT NUMBER
INTERNATIONAL CLASSIFICATION
ISSUE CLASSIFICATION
PTO 270
(REV. 6-81)

US005694593A

United States Patent [19]

Baclawski

[11] Patent Number: 5,694,593

[45] Date of Patent: Dec. 2, 1997

[54] **DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD**

[75] Inventor: **Kenneth P. Baclawski**, Waltham, Mass.

[73] Assignee: **Northeastern University**, Boston, Mass.

[21] Appl. No.: 318,252

[22] Filed: Oct. 5, 1994

[51] Int. Cl.[6] G06F 17/30

[52] U.S. Cl. 395/605; 395/603; 395/602; 395/621; 395/672

[58] Field of Search 395/600, 602, 395/603, 605, 621, 672; 364/419.19, 200

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,811,199	3/1989	Kuechler et al.	364/200
5,006,978	4/1991	Neches	364/200
5,265,207	11/1993	Zak et al.	395/200
5,309,359	5/1994	Katz et al.	364/419.19

OTHER PUBLICATIONS

Chanrvedi, et al., "Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach", IEEE Transactions On Engineering Management, vol. 41, No. 2, May 1994.

Houtsma et al., "Parallel Hierarchical Evaluation of Transitive Closure Queries", IEEE Apr. 1991.

Baclawski, K., "High-Performance, Indexing and Retrieval for Object-Oriented Databases", College of Computer Science, Northeastern University, Apr. 1994.

Baclawski et al., "High-Performance, Distributed Information Retrieval", College of Computer Science, Tecnical Report NU-CCS-94-05, Northeastern University, Feb. 25, 1994.

Salton, G., "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer", Cornell University, Chapter 10, Dec. 1988.

Salton et al., "Automatic Structuring and Retrieval of Large Text Files", Communications of the ACM, vol. 37, No.2, Feb. 1994.

Baclawski et al., "A distibuted approach to high-performance information retrieval", Northeastern University, Mar., 1994.

Baclawski et al., "A unified approach to high-performance, vector-based information retrieval", Northeastern University, Mar. 21, 1994.

Baclawski et al., "KEYNET: An architecture and protocol for high-performance semantically rich information retrieval", Northeastern University, Apr. 29, 1994.

Baclawski et al., "An abstract for sematically rich information retrieval", Northeastern University, Mar. 31, 1994.

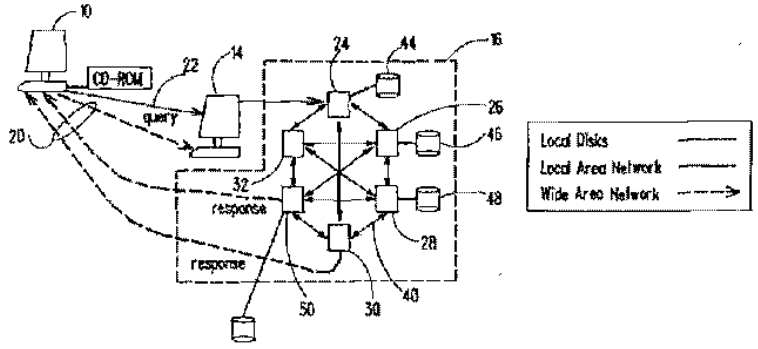
Baclawski et al., "KEYNET: Fast Indexing for semantically rich information retrieval", Northeastern University, Technical Report NU-CCS-94-06, Dec. 7, 1993.

Primary Examiner—Thomas G. Black
Assistant Examiner—Cheryl R. Lewis
Attorney, Agent, or Firm—Wolf, Greenfield & Sacks, P.C.

[57] **ABSTRACT**

A distributed computer database system including a front end computer and a plurality of computer nodes interconnected by a network into a search engine. A query from a user is transmitted to the front end computer which forwards the query to one of the computer nodes, termed the home node, of the search engine. The home node fragments the query and hashes the fragments of query to create an index by which the hashed query fragments are transmitted to one or more nodes on the network. Each node on the network which receives a hashed fragment uses the fragment of the query to perform a search on its respective database. The results of the searches of the local databases are then gathered by the home node.

17 Claims, 13 Drawing Sheets



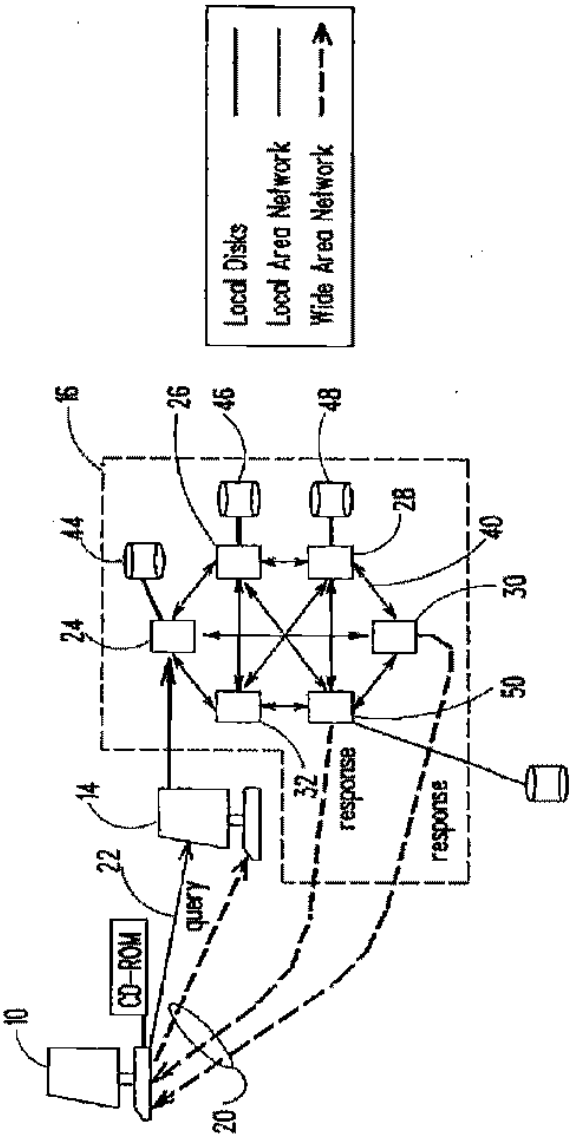

Local Disks
Local Area Network
Wide Area Network
16
26
46
48
44
28
40
24
30
50
32
response
response
14
22
query
CD-ROM
10
20

FIG. 1

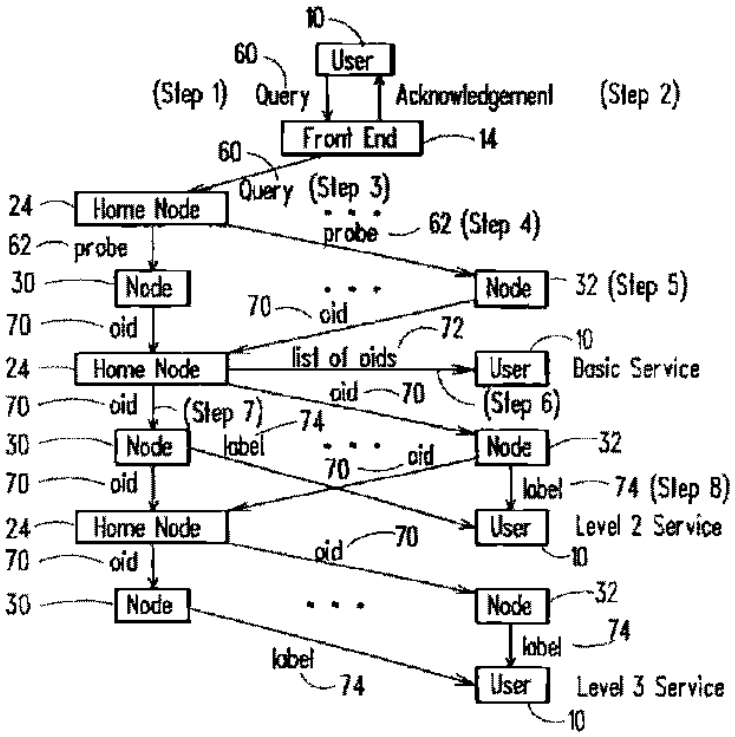

10
User
60
(Step 1) Query
Acknowledgement (Step 2)
Front End
14
60
Query (Step 3)
24
Home Node
probe
62 (Step 4)
62
probe
30
Node
Node
32 (Step 5)
70
oid
70
oid
72
24
Home Node
list of oids
User
10
Basic Service
70
oid
(Step 7)
oid
70
(Step 6)
74
30
Node
label
70
oid
Node
32
70
oid
label
74 (Step 8)
24
Home Node
User
Level 2 Service
70
oid
oid
70
10
30
Node
Node
32
label
74
label
74
User
Level 3 Service
10

FIG. 2

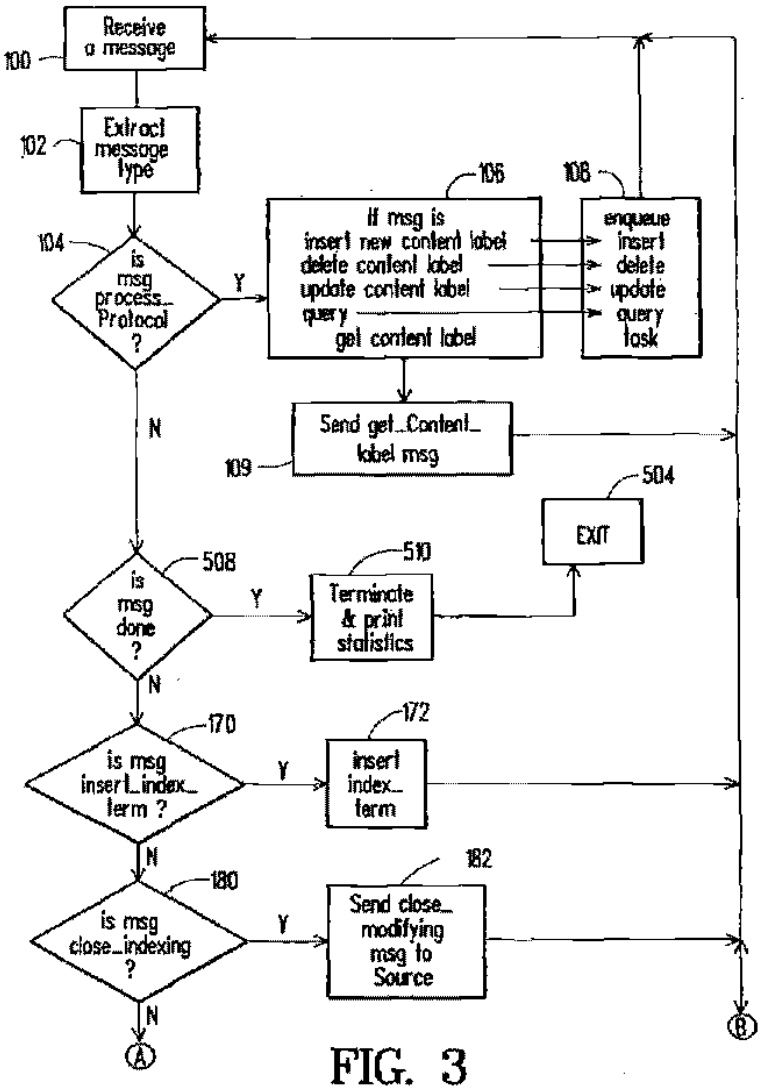

Receive a message
100
Extract message type
102
104
is msg process_ Protocol ?
Y
N
106
If msg is
insert new content label
delete content label
update content label
query
get content label
108
enqueue
insert
delete
update
query
task
Send get_Content_ label msg
109
504
EXIT
508
is msg done ?
Y
510
Terminate & print statistics
N
170
is msg insert_index_ term ?
Y
172
insert index_ term
N
180
is msg close_indexing ?
Y
182
Send close_ modifying msg to Source
N
A
B

FIG. 3

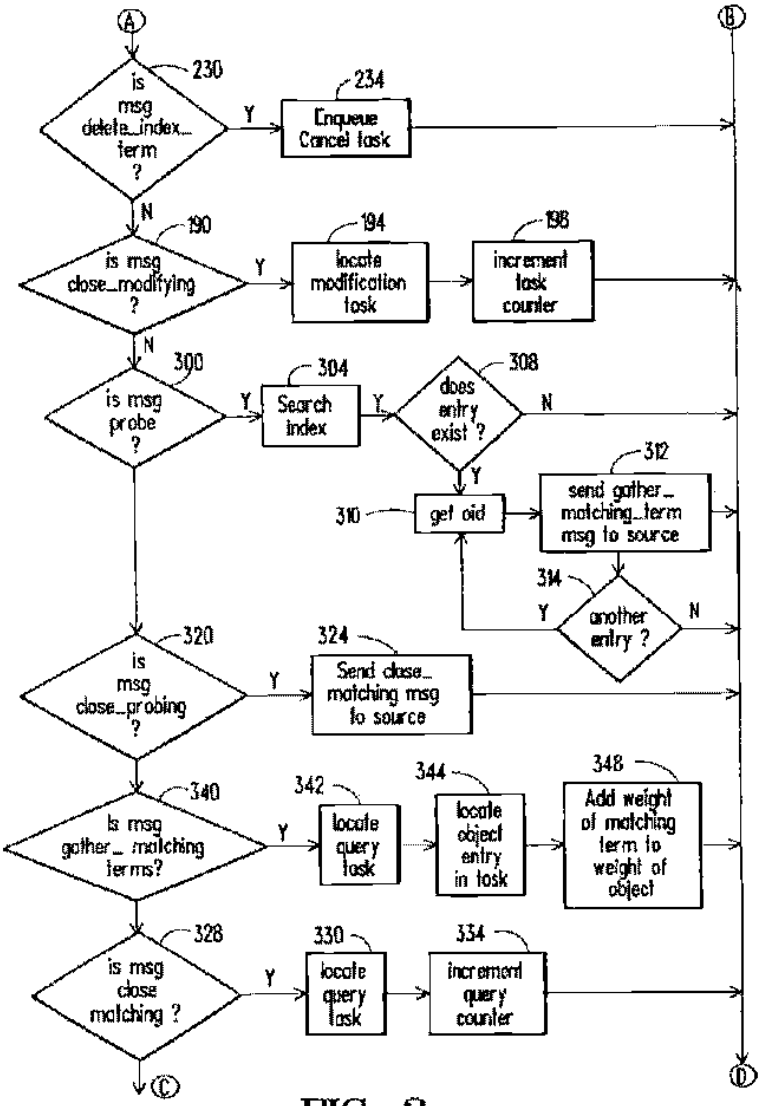

A
B
230
is msg delete_index_ term ?
Y
234
Enqueue Cancel task
N
190
is msg close_modifying ?
Y
194
locate modification task
198
increment task counter
N
300
is msg probe ?
Y
304
Search index
Y
308
does entry exist ?
N
Y
310
get oid
312
send gather_ matching_term msg to source
314
another entry ?
Y
N
320
is msg close_probing ?
Y
324
Send close_ matching msg to source
340
Is msg gather_ matching terms?
Y
342
locate query task
344
locate object entry in task
348
Add weight of matching term to weight of object
328
is msg close matching ?
Y
330
locate query task
334
increment query counter
C
D

FIG. 3a

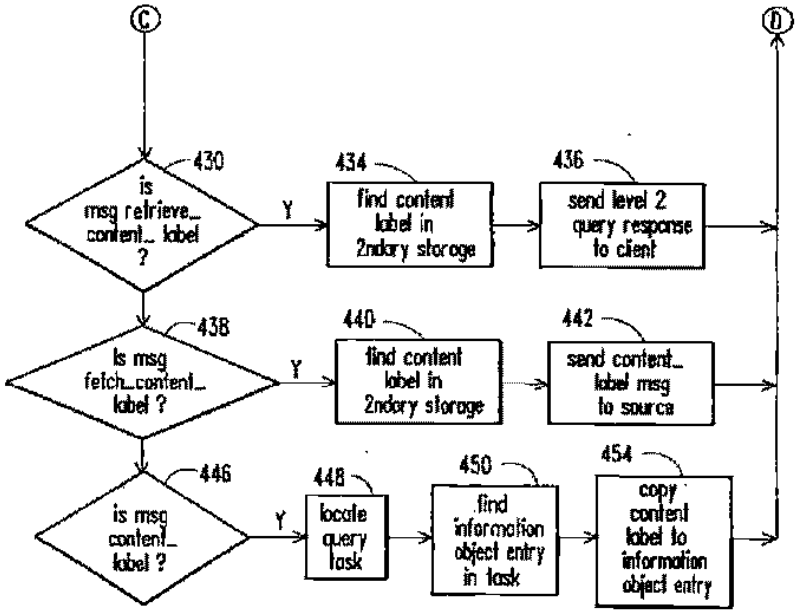

C
D
430
is msg retrieve_ content_ label ?
Y
434
find content label in 2ndary storage
436
send level 2 query response to client
438
Is msg fetch_content_ label ?
Y
440
find content label in 2ndary storage
442
send content_ label msg to source
446
is msg content_ label ?
Y
448
locate query task
450
find information object entry in task
454
copy content label to information object entry

FIG. 3b

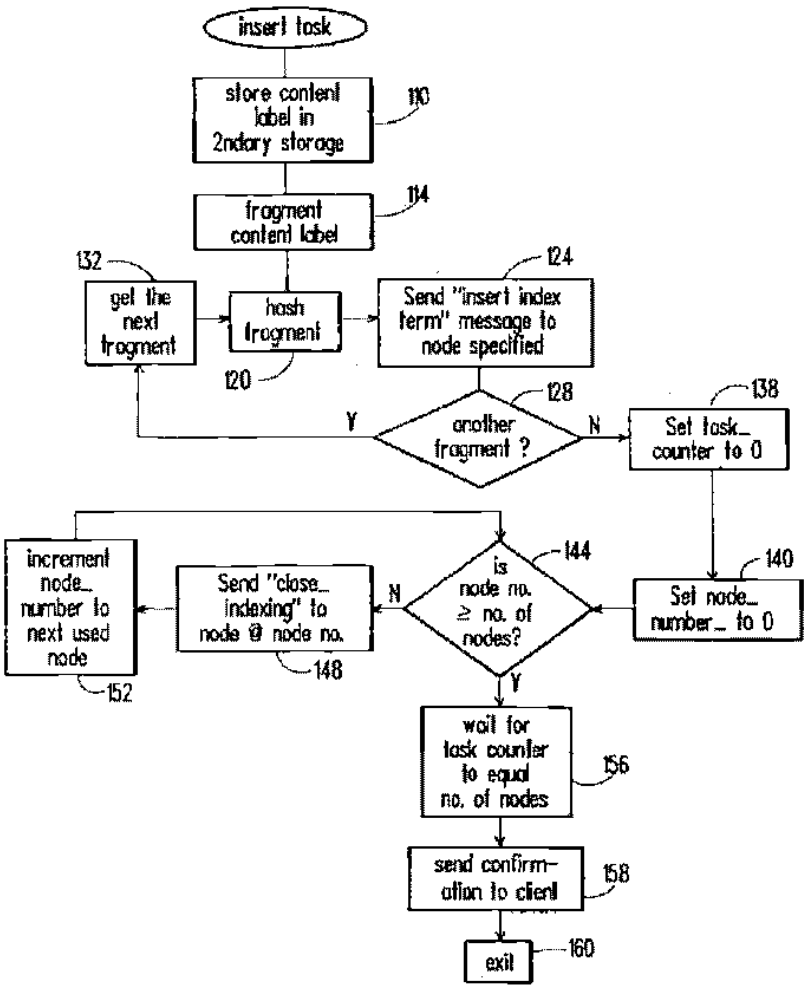

insert task
store content label in 2ndary storage
110
fragment content label
114
132
get the next fragment
hash fragment
120
124
Send "insert index term" message to node specified
128
another fragment ?
Y
N
138
Set task_ counter to 0
140
Set node_ number_ to 0
144
is node no. ≥ no. of nodes?
N
Y
Send "close_ indexing" to node @ node no.
148
increment node_ number to next used node
152
wait for task counter to equal no. of nodes
156
send confirm- ation to client
158
exit
160

FIG. 4

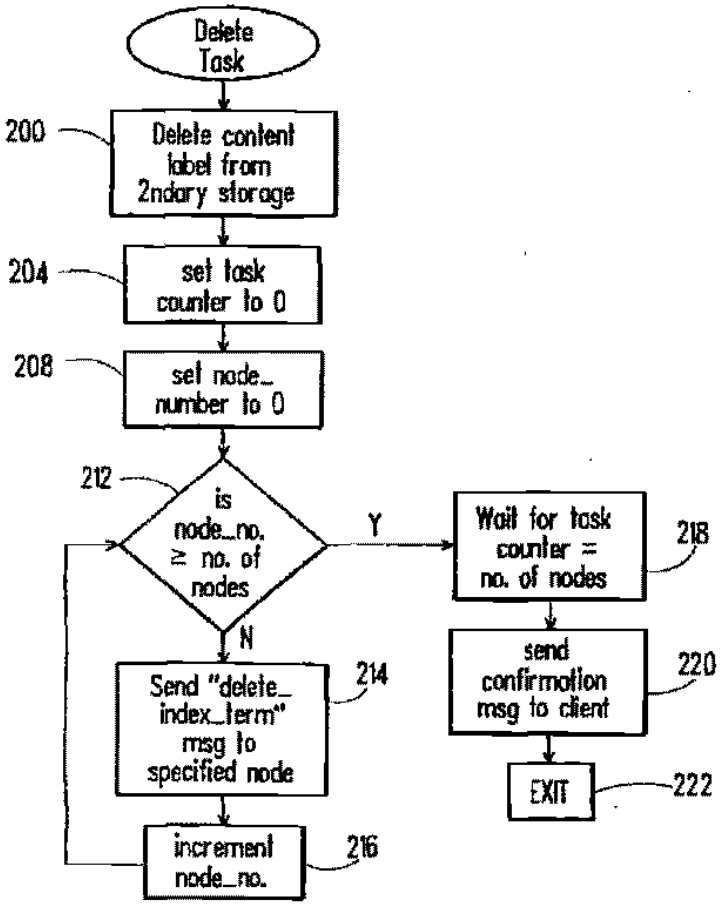

Delete Task
200
Delete content label from 2ndary storage
204
set task counter to 0
208
set node_ number to 0
212
is node_no. ≥ no. of nodes
Y
N
Wait for task counter = no. of nodes
218
send confirmation msg to client
220
EXIT
222
Send "delete_ index_term" msg to specified node
214
increment node_no.
216

FIG. 5

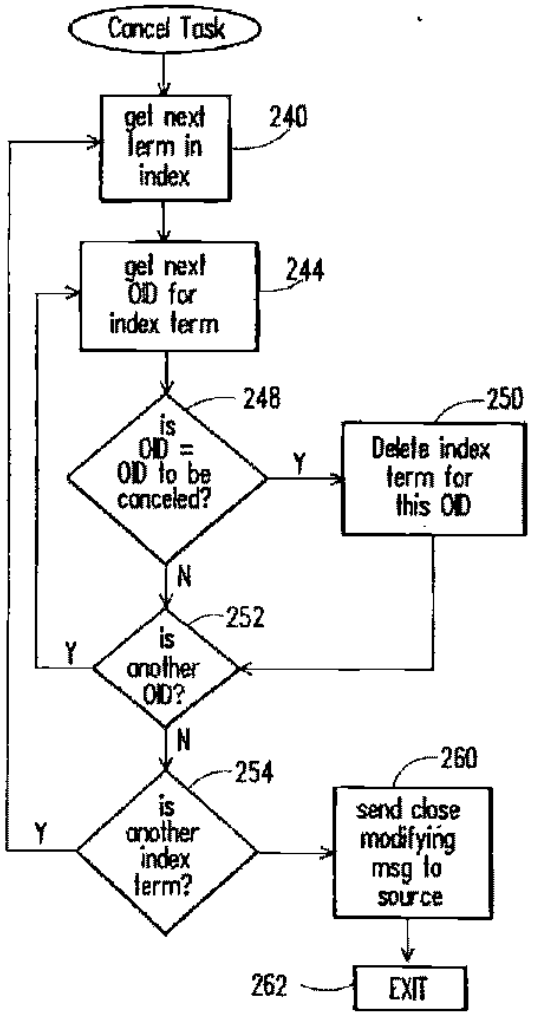

Cancel Task
get next term in index
240
get next OID for index term
244
248
is OID = OID to be canceled?
Y
250
Delete index term for this OID
N
252
is another OID?
Y
N
254
is another index term?
Y
260
send close modifying msg to source
262
EXIT

FIG. 6

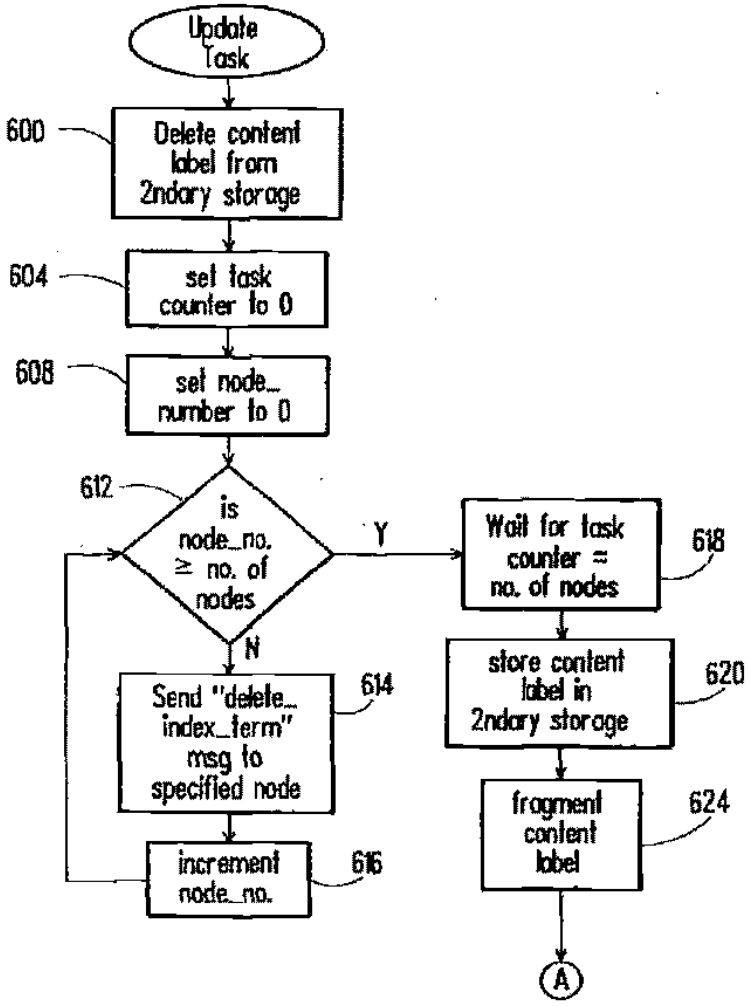

Update Task
600
Delete content label from 2ndary storage
604
set task counter to 0
608
set node_ number to 0
612
is node_no. ≥ no. of nodes
Y
N
Wait for task counter = no. of nodes
618
store content label in 2ndary storage
620
fragment content label
624
A
Send "delete_ index_term" msg to specified node
614
increment node_no.
616

FIG. 7

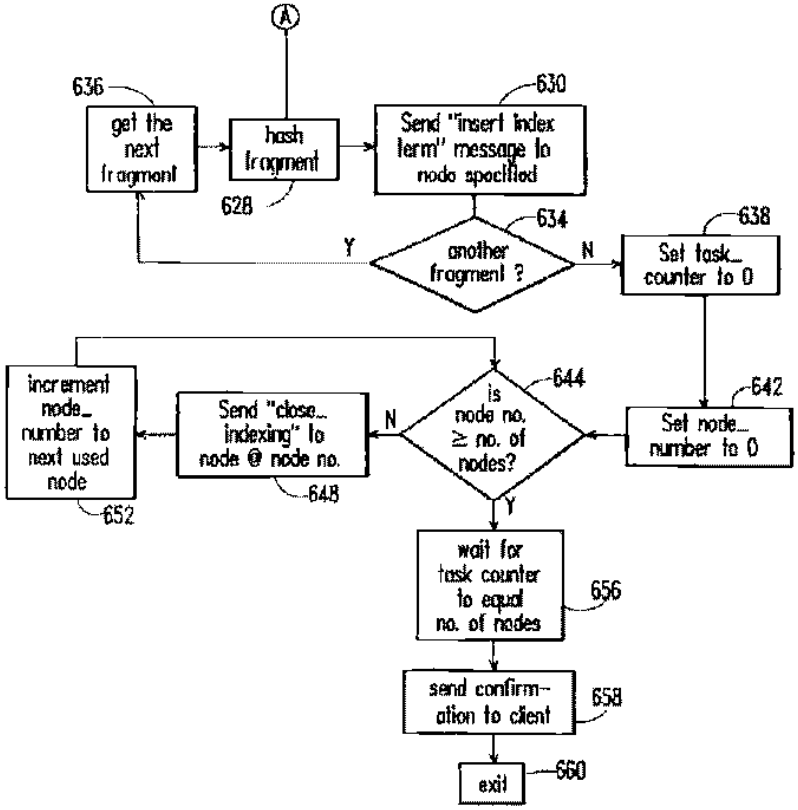

A
636
get the next fragment
hash fragment
628
630
Send "insert index term" message to node specified
634
another fragment ?
Y
N
638
Set task_ counter to 0
642
Set node_ number to 0
644
is node no. ≥ no. of nodes?
N
Y
Send "close_ indexing" to node @ node no.
648
increment node_ number to next used node
652
wait for task counter to equal no. of nodes
656
send confirm- ation to client
658
exit
660

FIG. 7a

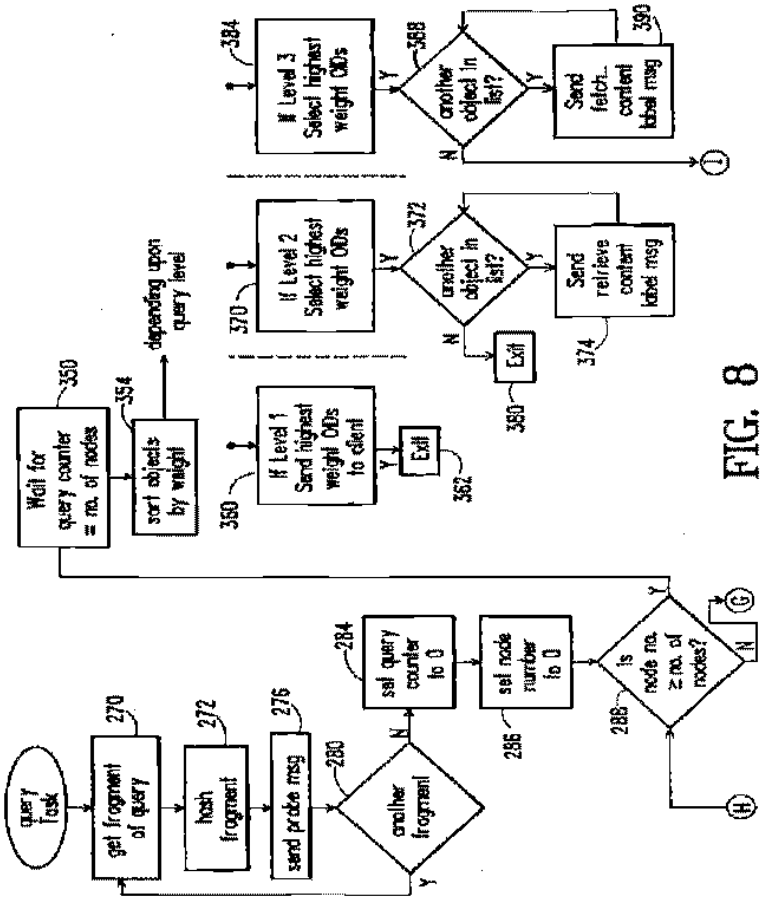

query task
get fragment of query
270
hash fragment
272
send probe msg
276
another fragment?
280
set query counter to 0
284
set node number to 0
286
is node no. ≥ no. of nodes?
288
Wait for query counter = no. of nodes
350
sort objects by weight
354
depending upon query level
If Level 1 Send highest weight OIDs to client
360
Exit
362
If Level 2 Select highest weight OIDs
370
another object in list?
372
Send retrieve content label msg
374
Exit
380
If Level 3 Select highest weight OIDs
384
another object in list?
388
Send fetch content label msg
390

FIG. 8

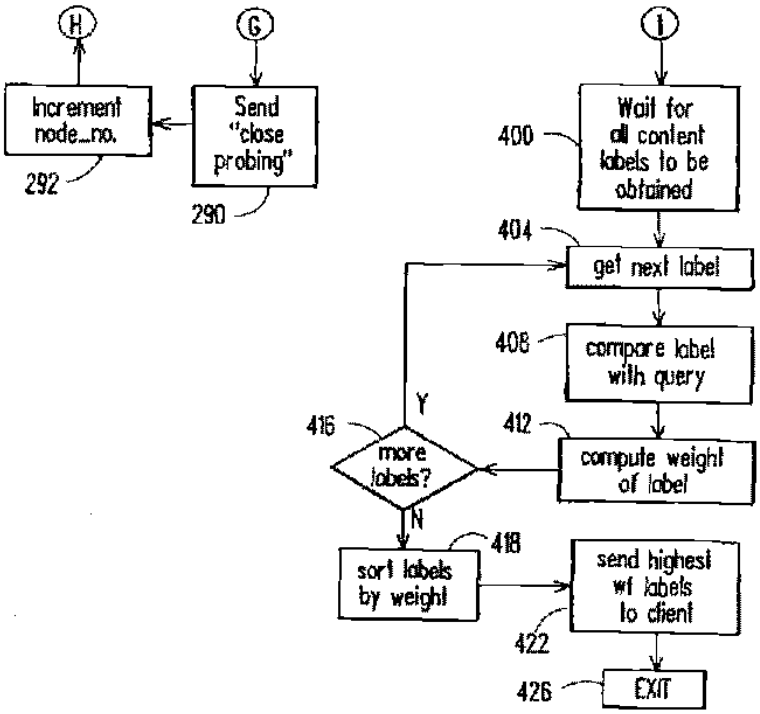

H
G
I
Increment node_no.
292
Send "close probing"
290
Wait for all content labels to be obtained
400
404
get next label
408
compare label with query
412
compute weight of label
416
more labels?
Y
N
418
sort labels by weight
send highest wt labels to client
422
426
EXIT

FIG. 8a

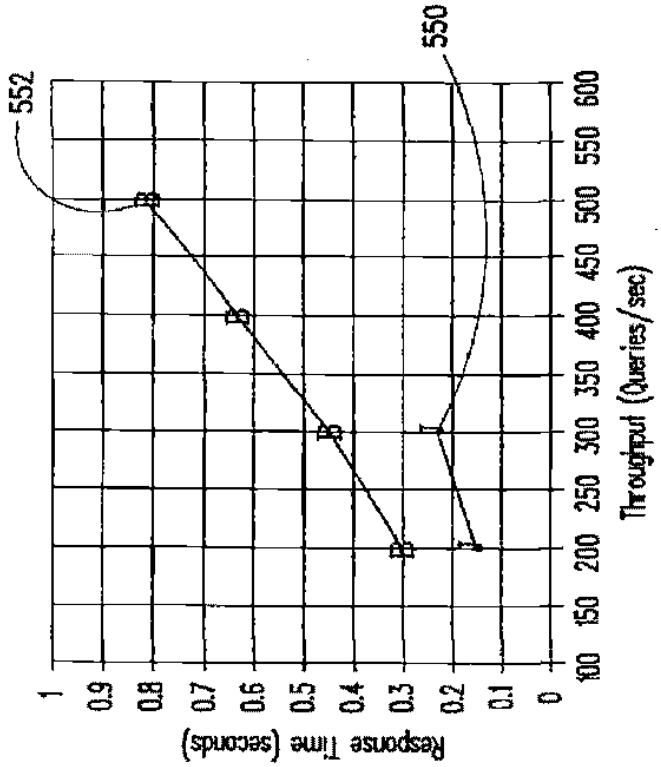

552
550
Response Time (seconds)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
150
200
250
300
350
400
450
500
550
600
Throughput (Queries/sec)

FIG. 9

DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

FIELD OF THE INVENTION

The invention relates to computer database systems and more specifically to distributed computer database systems.

BACKGROUND OF THE INVENTION

An object database consists of a collection of data or information objects. Each information object is identified uniquely by an object identifier (OID) and is described by a content label. The content label is written in a formal artificial language specified by the ontology of the database. The ontology specifies the data types, the access points or attributes of the data, the access or attribute values of the data, the join conditions or linking relationships between the data, and the grammar rules or constraints that must be satisfied by all content labels.

An ontology can also specify weight information such as the strength of a relationship or the degree of prototypicality of an attribute value. Weight information can also be included in content labels to distinguish more important parts of a content label from less important parts.

Queries to extract data from the database are written in the same formal language as the one used for content labels and hence must conform to the same ontology. A fragment of a content label or a query is a part of the content label or query consisting of a limited number of attributes and attribute values joined by relationships. An index term is a fragment of a content label, while a probe is a fragment of a query.

It should also be noted that the information object itself need not be stored in the database system as long as pointers to the data are available. Databases include indexes by which the database locates stored data. Large databases require correspondingly large indexes to maintain pointers to the stored data. Such an index can be larger than the database itself. Such large indexes are stored in relatively slow secondary storage rather than faster main memory because the main memory controlled by a single computer processor is limited in size.

Furthermore, each index term can be used to search for only one attribute of the data. Therefore, queries involving several attributes of the data are restricted to using only one index term corresponding to one attribute to locate the data. The remaining attributes of the data requested by the query are sequentially searched for, even if other indexes are available for the remaining attributes of the query. Additionally, a query that links or joins several attributes is performed using a sequential scan.

Finally, there is considerable overhead associated with maintaining an index. This limits the number of attributes that can be indexed. Current systems are unable to scale up to support databases for which there are: hundreds of data types; thousands of attributes; thousands of join conditions; queries that involve many data types, attributes and join conditions simultaneously; tens of millions of information objects; tens of millions of queries per day; rapid response-time-to-query requirements; and new data types, attributes and join conditions continually being added.

The present invention avoids these limitations.

SUMMARY OF THE INVENTION

The invention relates to a distributed computer database system which includes a front end computer and a plurality of computer nodes interconnected by a network. The combination of computer nodes interconnected by the network operates as a search engine.

A user wishing to query the database, transmits the query to the front end computer which in turn forwards the query to one of the computer nodes of the network. The node receiving the query, termed the home node of the search engine, fragments the received query and then hashes the fragments of the query. A portion of the hashed fragment is used by the home node as an addressing index by which the home node transmits the hashed query fragment to a node on the network.

Each node on the network which receives a hashed query fragment uses the fragment of the query to perform a search on its respective database. Nodes finding data corresponding to the query fragment return an identifier permitting the user to access the data in the database. Such identifiers are then gathered by the home node.

DESCRIPTION OF THE DRAWINGS

This invention is pointed out with particularity in the appended claims. The above and further advantages of this invention may be better understood by referring to the following description taken in conjunction with the accompanying drawing, in which:

FIG. 1 is a block diagram of an overview of an embodiment of the distributed computer database system of the invention;

FIG. 2 is an overview of the steps used by the embodiment of the distributed computer database system of claim 1 to respond to a query;

FIGS. 3, 3a and 3b are a functional diagram of an embodiment of a main message handling routine executed by the nodes of the embodiment of the distributed computer database system shown in FIG. 1;

FIG. 4 is a functional diagram of an embodiment of an INSERT task executed on the home node of the embodiment of the distributed computer database system shown in FIG. 1 to insert an index term into a local database on a node of the system;

FIG. 5 is a functional diagram of an embodiment of a DELETE task executed on the home node of the embodiment of the distributed computer database system shown in FIG. 1 to delete an index term;

FIG. 6 is a functional diagram of an embodiment of a CANCEL task executed on a query node of the embodiment of the distributed computer database system shown in FIG. 1 in response to a command from the home node to delete an index term from a local database;

FIGS. 7 and 7a is a functional diagram of an embodiment of an UPDATE task executed on the home node of the embodiment of the distributed computer database system shown in FIG. 1 to update an index term;

FIGS. 8 and 8a are a functional diagram of an embodiment of a QUERY task executed on the home node of the embodiment of the distributed computer database system shown in FIG. 1 to search for a content label; and

FIG. 9 is a graph of the 95th percentile response time to a query plotted against the throughput for the embodiment of the distributed computer database system shown in FIG. 1.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. 1, in broad overview, one embodiment of a distributed computer database system of the invention

includes a user computer **10** which is in communication with a front end computer **14** through a wide area network **20** or a local network **22**. The front end computer **14**, which may also be the user computer **10**, is in turn in communication with a search engine **16** which includes one or more computer nodes **24**, **26**, **28**, **30**, **32**, **50** interconnected by a local area network **40**. The individual computer nodes **24**, **26**, **28**, may include local disks **44**, **46**, **48** or may, alternatively or additionally, obtain data from a network disk server **50**.

In one embodiment each computer node **24**, **26**, **28**, **30**, **32** is a Sparcstation 10/30 with 32 MBytes of random access memory (RAM). The computer nodes are interconnected by a twisted pair network having a maximum data transfer rate of 100 Mbits/sec. In an alternative embodiment, all the computer nodes **24**, **26**, **28**, **30**, 32 are associated with local disks.

Considering the processing of a query first, and referring also to FIG. **2**, in one embodiment when a user transmits (Step 1) a query **60** from the user computer **10**, the front end computer **14** receives the query **60** and immediately issues an acknowledgement (Step 2). The front end computer **14** then transmits (Step 3) the query **60** to one of the computer nodes **24**, which is then defined as the home node **24** of the search engine **16** for that query **60**.

The home node **24** divides the query **60** into a number of (possibly overlapping) fragments or probes **62** which it then hashes using a predefined hashing function. Data in the system was previously stored locally on the various nodes using this hashing function to generate an index to the data in the local database. Thus, the use of the same hashing function to generate an index for data storage and to generate a hashed probe for data query assures that 1.) data is distributed uniformly over the nodes of the search engine during the storing of data and 2.) the probes are scattered uniformly over the nodes during the processing of a query.

In one embodiment, the hash value resulting from the use of the hashing function has a first portion which serves to identify the node to which the data is to be sent to be stored or to which query fragment is to be sent as a probe and a second portion which is the local index by which the data is stored at that node. In one embodiment, the hashing function reduces a query fragment to a 37 bit value, in which the first 5 bits designate the node to which the data or query is sent and the low order 32 bits which provides the index into the *local hash table* of the node's database. Thus, in terms of a query, the hashed query fragments are distributed (Step 4) as probes **62** to certain computer nodes **24**, **28**, **30**, **32**, of the search engine **16**, as determined by the first portion of the hash value.

At a first or basic service level, computer nodes **30**, **32** whose probes **62** match the index terms or labels by which the data was initially stored on that node respond to the query **60** by transmitting (Step 5) the object identifiers (OIDs) **70** matching the index terms of the requested information to the home node **24**. Thus, all matches between the hashed probes and the local hash table of index terms are returned or gathered to the home node which initially hashed the query.

Since the use of a hashing function results in non-relevant material also being accessed, the relevance of each object returned in the search must be determined. This determination of relevance is made by the home node **24** by comparing the degree of similarity between the query **60** and the information or object label **74** returned. In one embodiment the measure of similarity between the query **60** and the object label **74** is a cosine measure and is given by the expression COS (v,w), where the vector v denotes the query **60** and the vector w denotes the content label **74**. In one embodiment the N objects with the highest similarity are returned. In another embodiment all objects labels **74** which generate cosine values greater than a predetermined value are considered sufficiently similar to the query **60** to be returned to the user **10** as relevant information.

Once the similarity is determined, the home node **24** orders the OIDs according to their degree of similarity and then returns a list **72** of the most relevant OIDs **70** directly to the user computer **10** or any other computer specified (Step 6), by way of a wide area network **20** or local area network **20** without the intervention of the front end computer **14**.

Alternatively, for higher levels of service (level 2 and level 3), the home node **24** passes the most relevant OIDs **70** back to the computer nodes **28**, 32 (Step 7) which hold the information or content labels **74** identified by the OIDs **70**. These computer nodes **28**, 32 then pass the requested information or content labels **74** directly to the user **10** (Step 8).

In more detail, the highest priority task being executed on all the nodes of the search engine **16** is the message handling task shown in FIGS. 3, 3a and 3b. Each node waits to receive a message (**100**) which may be exchanged between the front end node **14** and the home node **24**; the home node **24** and the query nodes **26**, **28**, **30**, 32; the home node **24** and the client **10**; and the query nodes **26**, **28**, **30**, 32, and the client **10**. The message includes a TYPE field which identifies the type of message that is being sent, the source and destination addresses of the nodes, the data to be acted upon and weight information. Upon reception of a message the receiving node determines what type of message has been received by first extracting the message TYPE from the message packet (**102**).

If the message type received is a PROCESS—PROTOCOL message (**104**), the message has been sent by the front end node in response to a command from the user **10** and received by a node which is now designated the home node **24**. The PROCESS—PROTOCOL message therefore contains a command from the user **10** which is transmitted to the home node **24** by the front end node **14**. In one embodiment there are five valid commands (**106**): INSERT—NEW—CONTENT—LABEL, DELETE—CONTENT—LABEL, UPDATE—CONTENT—LABEL, QUERY and GET—CONTENT—LABEL. The home node **24** enqueues (**108**) an INSERT—TASK, a DELETE—TASK, an UPDATE—TASK or a QUERY—TASK, respectively, in response to the command INSERT—NEW—CONTENT—LABEL, DELETE—CONTENT—LABEL, UPDATE—CONTENT—LABEL, or QUERY contained in the message packet. Once the tasks are enqueued, they are executed as they reach the head of the queue.

In response to the GET—CONTENT—LABEL command, the home node **24** does not enqueue a task, but simply sends a RETRIEVE—CONTENT—LABEL message (**109**) to a query node, as is described in detail below.

Referring also to FIG. 4, if the command is an INSERT—NEW—CONTENT—LABEL, the home node **24** executes an INSERT—TASK. The INSERT—TASK stores the new content label in secondary storage (**110**) and then fragments the content label (**114**). The fragments of the content label are then hashed (**120**) into a hash value having a node number portion and a hash index value. The home node **24** then sends an INSERT—INDEX—TERM message contain-

ing the hash index value to the query node 26 specified by the node number portion of the hashed fragment (126). The home node 24 then determines if there is another fragment (128) and if so, retrieves (132) and hashes the next fragment (126). Once all the hashed fragments have been sent to the query nodes 26, 28 the home node 24 sets a TASK—COUNTER variable (138) and NODE—NUMBER variable to zero (140). The NODE—NUMBER variable is compared to the number of nodes in the network (144) and if the variable is less than the number of nodes on the network, the home node 24 sends a CLOSE—INDEXING message to the node 26 designated by the NODE—NUMBER variable (148) and increments the NODE—NUMBER variable to the next query node 28 in a list of nodes on the network (152). In another embodiment the home node 24 only increments the NODE—NUMBER variable to the next query node 28 to which at least one INSERT—INDEX—TERM message was sent. The CLOSE—INDEXING message instructs the query nodes 26, 28 that all the hashed index terms have been distributed. If the NODE—NUMBER variable is greater than or equal to the number of nodes in the network, the home node 24 waits for TASK—COUNTER (whose function is described below) to equal the number of nodes in the network (156), and once this occurs, sends a confirmation to the user 10 (158) of the completion of the index insertion task and then exits (160). In another embodiment the home node 24 only waits for the TASK—COUNTER to equal the number of nodes to which at least one INSERT—INDEX—TERM message was sent.

When a query node 26, 28 receives an INSERT—INDEX—TERM message (170) FIG. 3, the query node 26, 28 inserts the hashed index term into its local index table (172) using any of the algorithms known to those skilled in the art and loops to receive the next message (100). When the query node 26, 28 receives a CLOSE—INDEXING message, indicating that all the new index terms have been distributed, the query node 26, 28 replies with a CLOSE—MODIFYING message to the source or home node 24 (182) and again loops for the next message (100).

When the source or home node 24 receives a CLOSE—MODIFYING message (190), the home node 24 locates the task which was enqueued and generated the CLOSE—INDEXING message to which the query node is responding (194) and increments the TASK—COUNTER associated with that task (198). As shown in FIG. 4, it is upon the incrementing of the TASK—COUNTER that the INSERT—TASK is waiting (156). Once the TASK—COUNTER is incremented (198) the home node 24 loops to receive another message (100).

Referring again to FIG. 3, if the command is a DELETE—CONTENT—LABEL, the home node 24 enqueues and executes a DELETE TASK. The DELETE—TASK (FIG. 5) first deletes the content label from secondary storage (200). The variables TASK—COUNTER (204) and NODE—NUMBER (208) are then set to zero. The home node 24 then determines if the variable NODE—NUMBER is less than the number of nodes on the network (212) and if so, sends a DELETE—INDEX—TERM message to the node designated by the current NODE—NUMBER (214). The home node 24 then increments the variable NODE—NUMBER (216) and loops. If the variable NODE—NUMBER is greater than or equal to the number of nodes on the network, the home node 24 waits for the variable TASK—COUNTER to equal the number of nodes (218) and then sends a confirmation message to the client 10 that the DELETE—TASK has completed (220) prior to exiting (222).

Referring to FIG. 3a, when the query node 26, 28 receives the DELETE—INDEX—TERM, it enqueues a CANCEL—TASK to remove all the entries in the local database corresponding to the index term to be deleted (234) and loops to receive another message (100). Referring to FIG. 6, the CANCEL—TASK begins by reading an index term in the local index (240) and finding the OID which corresponds to that term (244). The query node 26,28 performs a sequential search of all OIDs in the hash table. Sequential scanning of the hash table is used because, in this embodiment, the original content label being delete is not known, so it cannot be fragmented and hashed as it is for the INSERT task. In other embodiments, it is possible to use the original content label and is desirable to do so if content labels are frequently updated. A determination is then made as to whether the OID is equal to the OID being deleted (248). If the OID is to be cancelled, the index term for this OID is deleted (250). A determination is then made as to whether there is another OID equal to the OID being deleted (252). If there is another OID equal to the OID being deleted, the index term for this OID is cancelled (250). If there is not another OID corresponding to the OID being deleted, a determination is made as to whether there is another index term (254) and if so, the next term in the index is read (240). If there are no more terms in the local hash table, the query node 26, 28 sends a CLOSE—MODIFYING message (260) to the source of the command or the home node 24 and exits (262).

As described with respect to the INSERT—INDEX—TERM command, (FIG. 3a) when the source or home node 24 receives a CLOSE—MODIFYING message (190), the home node 24 locates the task which was enqueued and which generated the DELETE—INDEX—TERM message to which the query node 26, 28 is responding (194) and increments the TASK—COUNTER associated with that task (198). It is upon this incrementing that the DELETE—TASK is waiting (218) FIG. 5. Once the TASK—COUNTER is incremented (198) the home node 24 loops to receive another message (100).

Referring again to FIG. 3, if the command is a UPDATE—CONTENT—LABEL (106), the home node 24 enqueues and executes an UPDATE—TASK (168). The UPDATE—TASK (FIGS. 7 and 7a) is a combination of an DELETE—TASK followed by an INSERT—TASK, without a confirmation being sent at the end of the DELETE—TASK. The UPDATE—TASK first deletes the content label from secondary storage (600) and then sets the variables TASK—COUNTER (604) and NODE—NUMBER (608) to zero. The home node 24 determines if the variable NODE—NUMBER is less than the number of nodes on the network (612) and if so, sends a DELETE—INDEX—TERM message to the node designated by the current NODE—NUMBER (614), increments the variable NODE—NUMBER (616) and loops. If the variable NODE—NUMBER is greater than or equal to the number of nodes on the network, the home node 24 waits for the variable TASK—COUNTER to equal the number of nodes (618). When TASK—COUNTER is equal to the number of nodes, the task stores the new content label in secondary storage (620) and then fragments the content label (624). As described previously, the fragments of the content label are then hashed (628) into a hash value having a node number portion and a hash index value. The home node 24 then sends an INSERT—INDEX—TERM message containing the hash index value to the query node 26 specified by the node number portion of the hashed fragment (630). The home node 24 then determines if there is another fragment (634) and if so, retrieves (636) and hashes the next fragment

(628). Once all the hashed fragments have been sent to the query nodes 26, the home node 24 sets a TASK—COUNTER variable (638) and NODE—NUMBER variable to zero (642). The NODE—NUMBER variable is compared to the number of nodes in the network (644) and if the variable is less than the number of nodes on the network, the home node 24 sends a CLOSE—INDEXING message to the node 26 designated by the NODE—NUMBER variable (648) and increments the NODE—NUMBER variable to the next query node 28 in a list of nodes on the network (652). As discussed with respect to the INSERT—TASK, in another embodiment the home node 24 only increments the NODE—NUMBER variable to the next query node 28 to which the INSERT—INDEX—TERM message was sent. If the NODE—NUMBER variable is greater than or equal to the number of nodes in the network, the home node 24 waits for TASK—COUNTER to equal the number of nodes in the network (656), and once this occurs, sends a confirmation to the user 10 (658) of the completion of the UPDATE—TASK and then exits (660). The response by the query node to the messages sent by the UPDATE—TASK have been described above with respect to the INSERT—TASK and the DELETE—TASK.

Referring to FIG. 3, if the command is a QUERY (106), the home node 24 enqueues and executes a QUERY—TASK (108) as described above. Referring also to FIGS. 5 and 5a, the home node 24 fragments the query 60 (270) and hashes the fragment (272) using the same hashing function that was used by the INSERT—TASK as disclosed above. The home node 24 then sends a PROBE message (276) to the node 26 whose address is indicated by the first portion of the hashed fragment. The PROBE message to the query node 26 includes the second portion of the hashed query fragment. The home node 24 then determines whether there is another fragment of the query to process (280). If the is another fragment, that fragment is obtained (270), hashed (272) and transmitted to the node indicated by the first portion of the hashed fragment (276).

If there are no further fragments to process, the home node 24 sets the TASK—COUNTER variable (284) and the NODE—NUMBER variable (286) to zero, and a determination is made whether the NODE—NUMBER is less than the number of nodes in the network (288). If NODE—NUMBER is less than the number of nodes, a CLOSE—PROBING message is sent (290) to the query node indicated by the NODE—NUMBER variable (to inform the query node that all the query fragments have been sent to the query nodes) and the NODE—NUMBER variable is incremented (292).

Referring again to FIG. 3a, when a query node 26 receives a PROBE message, the query node 26 searches its local index for a match to the second portion of the hashed fragment contained in the PROBE message (304). If no matching index term exists (308), the query node 26 simply awaits the next message (100). If a match exists, the OID having the hash value is accessed (316) and the query node 26 sends a GATHER—MATCHING—TERMS message back to the home node 24 (312). The query node 26 then determines if there is another entry in the index matching the hash value in the PROBE message (314). If another entry in the database is matched by the index term, the query node 26 again gets the OID (316) and sends a GATHER—MATCHING—TERMS message back to the home node (318). If no entries are matched by the index term, the query node 26 loops to receive the next message (100).

Similarly, when a query node 26 receives a CLOSE—PROBING message from the home node 24 (320), the query node 26 sends a CLOSE—MATCHING message to the source or home node 24 (324). When the home node 24 receives a CLOSE—MATCHING message from the query node 26 (328), it locates the QUERY task (330) to which the query node 26 is responding and increments the variable QUERY—COUNTER (334) and loops to receive another message (100).

When the home node 24 receives a GATHER—MATCHING—TERMS message from the query node 26 (340), the home node 24 locates (342) the QUERY task which generated the probe 62 which caused the GATHER—MATCHING—TERMS message to be sent and locates the object entry in the task (344). The home node 24 then adds the weight of the matching term to the total weight of the object (348) and returns to await another message (100).

In one embodiment, the weight of the matching term is computed using a COS function. The query and the index term are treated as vectors with the weight of the matching term determined by the COS function COS(v,w), where v is the query vector and w is the index term vector.

Similarly, the total weight of the object is determined from the COS function according to the following formula:

$$COS(v,w)=\frac{\sum_i v_i w_i}{\|v\|\|w\|}$$

The sum is over all terms (i.e. both index and query terms) and the product $v_i w_i$ is non-zero only if the term occurs both in the query and in the content label. The length of the vectors v and w are computed by the formulae:

$$\|v\|=\sqrt{\sum_i (v_i)^2}$$

$$\|w\|=\sqrt{\sum_i (w_i)^2}$$

If v represents the query, then $\|v\|$ is never actually computed since weights are only considered relative to one another for the same query. The value of $\|w\|$ is precomputed and what is actually stored are the normalized content label vectors $w/\|w\|$.

Referring again to FIG. 8, when the QUERY—COUNTER is equal to the number of nodes (350) all the requested data has been received and the object labels are sorted by weight to determine the closeness of the matching of the query 60 and the content labels 74 retrieved (354). The remaining instructions executed by the QUERY task, depend upon the query level. If the query level is one, only the OIDs with the highest weights are returned to the client 10 (360) and the QUERY task exits (362).

If the query level is two, the OIDs with the highest weights are selected and a determination is made as to which objects correspond to those highest weight OIDs (370). The highest weight content labels are selected (372) and the list of highest weight content labels 72 are sent by way of a RETRIEVE—CONTENT—LABEL message to the query node 26 (374) having a node number equal to the information object identifier modulo the number of nodes. That is, the OID modulo the number of nodes can be regarded as another hash function that maps the OID to the node containing the content label. The set of nodes that store content labels in secondary storage need not be the same set as the one that stores the hash table in main memory. The hash function for fragments maps a fragment to the set of

matching OIDs. The hash function for OIDs maps an OID to the node containing the corresponding content label. The content label contains information on how to acquire the information objects themselves. If no further objects exist, the QUERY task exits (380).

Referring also to FIG. 3b, when a query node 26 receives a RETRIEVE—CONTENT—LABEL (430), it locates the content label in secondary storage (434) and sends the level 2 query response to the client 10 (436). The query node 26 then loops for another message (100).

Referring again to FIG. 8a, as with level two, if the query level is three, the highest weight OIDs are selected, a determination is made as to which objects correspond to those highest weight OIDs (384) and those objects are selected (388). A FETCH—CONTENT—LABEL message is sent to the query node 26 (390) again corresponding to the information object identifier modulo the number of nodes.

Referring again to FIG. 3b, when the query node 26, receives a FETCH—CONTENT—LABEL message 438, the query node 26 finds the content label in secondary storage 440 and sends a CONTENT—LABEL message to the source of the FETCH—CONTENT—LABEL (442). When the home node 24, receives the CONTENT—LABEL message (446), it locates the QUERY task which generated the FETCH—CONTENT—LABEL message (448), finds the information object entry within the QUERY task (450) and copies the content label to the information object entry (454). The home node 24 then loops to receive another message (100).

Referring again to FIG. 8a, if no further objects exist in the list, the QUERY task waits for all the content labels to be obtained (400). Once all have been obtained, the QUERY task obtains a content label (404) and compares the content label with the query 60 (408). The QUERY task then computes the weight of the label (412). Unlike the weight computed in level 1 and 2 processing, which use fragments of a limited size, the weight computed in level 3 processing considers fragments of any size. In addition, the fragments of the content label that match fragments of the query are marked in the content label. Therefore, if the content label is shown to the user who requested the query, the marked fragments can be distinguished, for example by highlighting, to show the user the reason why the content label was chosen in response to the user's query. Highlighting also serves to focus the user's attention on those portions of the content label that are most likely to be relevant and aid in helping the user formulate subsequent queries. A determination is then made as to whether there are more content labels to process (416) and if so, the next content label is fetched (404). If there are no further content labels, the labels are sorted by weight (418) and the content labels with the highest weights are sent to the client 10 (422). The QUERY task then exits (426).

Referring to FIG. 3, the command DONE is issued to terminate network activity, for example, to permit the software or ontology to be modified. If the message received from the front end node 14 is DONE 508, the activity on the network is terminated and is accompanied by the gathering and printing of activity statistics by the home node 24 (510). When the statistics are printed, the task exits (504).

A graph showing the 95th percentile response time versus throughput (number of queries processed per second) for a four node 550 and an eight node 552 embodiment of system of FIG. 1 is shown in FIG. 9. In this example the database consists of 80,000 information objects indexed by four nodes and 160,000 information objects indexed by eight nodes. The content labels have, on average, 200 index terms, and the queries have, on average, 10 fragments. Each probe produces, on average, four GATHER—MATCHING—TERMS messages at the query node.

It should be noted that tests on a 16-node system have resulted in the same response time graph as that for an 8-node system. Careful monitoring of the system has shown that statistical fluctuations dissipate quickly and the system is very stable as long as the bandwidth of the network is not approached too closely. The network vendor, for the network used in the embodiment disclosed, recommends that the network not be used for transmitting more than 40% of the rated maximum bandwidth. Thus for the embodiment disclosed, this limit is about 500 Kbytes/sec (4 Mbits/second). It should be noted, the actual response time for any system is very sensitive to the properties of the database and the distribution of queries. Thus, the throughput achieved by any given system may vary.

Having shown the preferred embodiment, those skilled in the art will realize many variations are possible which will still be within the scope and spirit of the claimed invention. Therefore,

It is the intention to limit the invention only as indicated by the scope of the claims.

What is claimed is:

1. A method for information retrieval using fuzzy queries in a non-relational, distributed database system having a plurality of home nodes and a plurality of query nodes connected by a network, said method comprising the steps of:

randomly selecting a first one of said plurality of home nodes;

fragmenting, by said selected home node, a query from a user into a plurality of query fragments;

hashing, by said selected home node, each said query fragment of said plurality of query fragments, said hashed query fragment having a first portion and a second portion;

transmitting, by said selected home node, each said hashed query fragment of said plurality of query fragments to a respective one of said plurality of query nodes indicated by said first portion of each said hashed query fragment;

using, by said query node, said second portion of said respective hashed query fragment to access data according to a local hash table located on said query node; and

returning, by each said query node accessing data according to said respective hashed query fragment, an object identifier corresponding to said accessed data to said selected home node.

2. The method of claim 1 further comprising the step of receiving, at said home node, said query from said user, prior to the step of fragmenting said query.

3. The method of claim 1 further comprising the steps of:

determining, by said home node, a measure of relevance between said accessed data and said query; and

returning, to said user, by said home node, accessed data having a predetermined degree of relevance.

subsequent to the step of returning said object identifier.

4. The method of claim 3 wherein said measure of relevance is determined by a cosine measure.

5. The method of claim 1 wherein said first portion of said hashed query fragment comprises 5 bits and said second portion comprises 32 bits.

6. A method of storing objects in a manner which is conducive to information retrieval using fuzzy queries in a

non-relational, distributed database system having a plurality of home nodes and a plurality of query nodes connected by a network, said method comprising the steps of:

randomly selecting a first one of said plurality of home nodes;

fragmenting, by said selected home node, objects from a user into a plurality of object fragments;

hashing, by said selected home node, each said object fragment of said plurality of object fragments, said hashed object fragment having a first portion and a second portion;

transmitting, by said selected home node, each said hashed object fragment of said plurality of data fragments to a respective one of said plurality of query nodes indicated by said first portion of each said hashed object fragment; and

using, by said query node, said second portion of said respective hashed object fragment to store data according to a local hash table located on said query node.

7. The method of claim 6 further comprising the step of receiving, at said home node, said objects from said user, prior to the step of fragmenting said object.

8. A non-relational, distributed database system having an information retrieval tool for handling queries from a user, comprising:

a plurality of home nodes; and

a plurality of query nodes;

said plurality of home nodes and said plurality of query nodes connected by a network,

wherein each said home node, upon receiving a query from a user, fragments said query into a plurality of query fragments, hashes each said query fragment of said plurality of query fragments into a hashed query fragment having a first portion and a second portion, and transmits each said hashed query fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed query fragment, and

further wherein each said query node uses said second portion of said hashed query fragment to access data according to a local hash table located on said query node and returns an object identifier corresponding to said accessed data to said home node.

9. The distributed database system of claim 8 wherein said home node determines a measure of relevance between said accessed data and said query and returns to said user accessed data having a predetermined degree of relevance.

10. The method of claim 9 wherein said home node measures relevance using a cosine measure.

11. The method of claim 8 wherein said first portion of said hashed query fragment comprises 5 bits and said second portion comprises 32 bits.

12. A non-relational, distributed database system for storage and retrieval of information objects, comprising:

a plurality of home nodes; and

a plurality of query nodes;

said plurality of home nodes and said plurality of query nodes connected by a network,

wherein each said home node, upon receiving an object from a user, fragments said object into a plurality of object fragments, hashes each said object fragment of said plurality of object fragments into a hashed object fragment having a first portion and a second portion, and transmits each said hashed object fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed object fragment, and

wherein each said query node uses said second portion of said hashed object fragment to store objects according to a local hash table located on said query node.

13. A non-relational, distributed database system having an information retrieval tool for handling queries from a user, comprising:

a plurality of home nodes; and

a plurality of query nodes, said plurality of home nodes and said plurality of query nodes connected by a network,

each said home node, upon receiving a command from a user, enqueueing a predetermined task in response to said command,

a query task enqueued being resultant in, in response to a query command from said user, fragmenting a query contained in said query command into a plurality of query fragments, hashing each said query fragment of said plurality of query fragments into a hashed query fragment having a first portion and a second portion, and transmitting a query message containing each said hashed query fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed query fragment,

said query node, upon receipt of said query message, using said second portion of said hashed query fragment to access data according to a local hash table located on said query node and transmitting a message returning an object identifier corresponding to said accessed data to said home node.

14. The method of claim 13 wherein said query message requests predetermined data from said query node in response to a query level contained in said query command from said user.

15. The method of claim 14 wherein there are three query levels.

16. The method of claim 14 wherein said query node returns a content label in response to a predetermined query level.

17. A non-relational, distributed database system for storage and retrieval of information, comprising:

a plurality of home node nodes; and

a plurality of query nodes, said plurality of home nodes and said plurality of query nodes connected by a network,

each said home node, upon receiving a command from a user, enqueueing a predetermined task in response to said command,

an insert task enqueued, in response to an insert command from said user, fragmenting data contained in said insert command into a plurality of data fragments, hashing each said data fragment of said plurality of data fragments into a hashed data fragment having a first portion and a second portion, and transmitting an insert message containing each said hashed data fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed data fragment,

said query node, upon receipt of said insert message, using said second portion of said hashed data fragment to store data according to a local hash table located on said query node.

* * * * *

PATENT APPLICATION SERIAL NO 08/318252

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

050 FW 10/18/94 08318252 1 201 479.00 CK RU-360XX

PTO-1556
(5/87)

B

79.00 08/318252

Patent

DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

ABSTRACT

A distributed computer database system including a front end computer and a plurality of computer nodes interconnected by a network into a search engine. A query from a user is transmitted to the front end computer which forwards the query to one of the computer nodes, termed the home node, of the search engine. The home node fragments the query and hashes the fragments of query to create an index by which the hashed query fragments are transmitted to one or more nodes on the network. Each node on the network which receives a hashed fragment uses the fragment of the query to perform a search on its respective database. The results of the searches of the local databases are then gathered by the home node.

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

08/318,252

FIELD OF THE INVENTION

This invention relates to computer database systems and more specifically to distributed computer database systems.

BACKGROUND OF THE INVENTION

An object database consists of a collection of data or information objects. Each information object is identified uniquely by an object identifier (OID) and is described by a content label. The content label is written in a formal artificial language specified by the ontology of the database. The ontology specifies the data types, the access points or attributes of the data, the access or attribute values of the data, the join conditions or linking relationships between the data, and the grammar rules or constraints that must be satisfied by all content labels.

An ontology can also specify weight information such as the strength of a relationship or the degree of proto-typicality of an attribute value. Weight information can also be included in content labels to distinguish more important parts of a content label from less important parts.

Queries to extract data from the database are written in the same formal language as the one used for content labels and hence must conform to the same ontology. A fragment of a content label or a query is a part of the content label or query consisting of a limited number of attributes and attribute values joined by relationships. An index term is a fragment of a content label, while a probe is a fragment of a query.

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

It should also be noted that the information object itself need not be stored in the database system as long as pointers to the data are available. Databases include indexes by which the database locates stored data. Large databases require correspondingly large indexes to maintain pointers to the stored data. Such an index can be larger than the database itself. Such large indexes are stored in relatively slow secondary storage rather than faster main memory because the main memory controlled by a single computer processor is limited in size.

Furthermore, each index term can be used to search for only one attribute of the data. Therefore, queries involving several attributes of the data are restricted to using only one index term corresponding to one attribute to locate the data. The remaining attributes of the data requested by the query are sequentially searched for, even if other indexes are available for the remaining attributes of the query. Additionally, a query that links or joins several attributes is performed using a sequential scan.

Finally, there is considerable overhead associated with maintaining an index. This limits the number of attributes that can be indexed. Current systems are unable to scale up to support databases for which there are: hundreds of data types; thousands of attributes; thousands of join conditions; queries that involve many data types, attributes and join conditions simultaneously; tens of millions of information objects; tens of millions of queries per day; rapid response-time-to-query requirements; and new data types, attributes and join conditions continually being added.

The present invention avoids these limitations.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

SUMMARY OF THE INVENTION

The invention relates to a distributed computer database system which includes a front end computer and a plurality of computer nodes interconnected by a network. The combination of computer nodes interconnected by the network operates as a search engine.

A user wishing to query the database, transmits the query to the front end computer which in turn forwards the query to one of the computer nodes of the network. The node receiving the query, termed the home node of the search engine, fragments the received query and then hashes the fragments of the query. A portion of the hashed fragment is used by the home node as an addressing index by which the home node transmits the hashed query fragment to a node on the network.

Each node on the network which receives a hashed query fragment uses the fragment of the query to perform a search on its respective database. Nodes finding data corresponding to the query fragment return an identifier permitting the user to access the data in the database. Such identifiers are then gathered by the home node.

DESCRIPTION OF THE DRAWINGS

This invention is pointed out with particularity in the appended claims. The above and further advantages of this invention may be better understood by referring to the following description taken in conjunction with the accompanying drawing, in which:

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

Fig. 1 is a block diagram of an overview of an embodiment of the distributed computer database system of the invention;

Fig. 2 is an overview of the steps used by the embodiment of the distributed computer database system of claim 1 to respond to a query;

Figs. 3, 3a and 3b are a functional diagram of an embodiment of a main message handling routine executed by the nodes of the embodiment of the distributed computer database system shown in Fig. 1;

Fig. 4 is a functional diagram of an embodiment of an INSERT task executed on the home node of the embodiment of the distributed computer database system shown in Fig. 1 to insert an index term into a local database on a node of the system;

Fig. 5 is a functional diagram of an embodiment of a DELETE task executed on the home node of the embodiment of the distributed computer database system shown in Fig. 1 to delete an index term;

Fig. 6 is a functional diagram of an embodiment of a CANCEL task executed on a query node of the embodiment of the distributed computer database system shown in Fig. 1 in response to a command from the home node to delete an index term from a local database;

Figs. 7 and 7a is a functional diagram of an embodiment of an UPDATE task executed on the home node of the embodiment of the distributed computer database system shown in Fig. 1 to update an index term;

Figs. 8 and 8a are a functional diagram of an embodiment of a QUERY task executed on the home node of the embodiment of the

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

distributed computer database system shown in Fig. 1 to search for a content label; and

Fig. 9 is a graph of the 95* percentile response time to a query plotted against the throughput for the embodiment of the distributed computer database system shown in Fig. 1.

DETAILED DESCRIPTION OF THE INVENTION

Referring to Fig. 1, in broad overview, one embodiment of a distributed computer database system of the invention includes a user computer 10 which is in communication with a front end computer 14 through a wide area network 20 or a local network 22. The front end computer 14, which may also be the user computer 10, is in turn in communication with a search engine 16 which includes one or more computer nodes 24, 26, 28, 30, 32, 50 interconnected by a local area network 40. The individual computer nodes 24, 26, 28, may include local disks 44, 46, 48 or may, alternatively or additionally, obtain data from a network disk server 50.

In one embodiment each computer node 24, 26, 28, 30, 32 is a Sparcstation 10/30 with 32 MBytes of random access memory (RAM). The computer nodes are interconnected by a twisted pair network having a maximum data transfer rate of 100 Mbits/sec. In an alternative embodiment, all the computer nodes 24, 26, 28, 30, 32 are associated with local disks.

Considering the processing of a query first, and referring also to Fig. 2, in one embodiment when a user transmits (Step 1) a query 60 from the user computer 10, the front end computer 14 receives the query 60 and immediately issues an acknowledgement

FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER
TEL (617) 542-2290
FAX (617) 451-0313

(Step 2). The front end computer 14 then transmits (Step 3) the query 60 to one of the computer nodes 24, which is then defined as the home node 24 of the search engine 16 for that query 60.

The home node 24 divides the query 60 into a number of (possibly overlapping) fragments or probes 62 which it then hashes using a predefined hashing function. Data in the system was previously stored locally on the various nodes using this hashing function to generate an index to the data in the local database. Thus, the use of the same hashing function to generate an index for data storage and to generate a hashed probe for data query assures that 1.) data is distributed uniformly over the nodes of the search engine during the storing of data and 2.) the probes are scattered uniformly over the nodes during the processing of a query.

In one embodiment, the hash value resulting from the use of the hashing function has a first portion which serves to identify the node to which the data is to be sent to be stored or to which query fragment is to be sent as a probe and a second portion which is the local index by which the data is stored at that node. In one embodiment, the hashing function reduces a query fragment to a 37 bit value, in which the first 5 bits designate the node to which the data or query is sent and the low order 32 bits which provides the index into the local hash table of the node's database. Thus, in terms of a query, the hashed query fragments are distributed (Step 4) as probes 62 to certain computer nodes 26, 28, 30, 32, of the search engine 16, as determined by the first portion of the hash value.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

At a first or basic service level, computer nodes 30, 32 whose probes 62 match the index terms or labels by which the data was initially stored on that node respond to the query 60 by transmitting (Step 5) the object identifiers (OIDs) 70 matching the index terms of the requested information to the home node 24. Thus, all matches between the hashed probes and the local hash table of index terms are returned or gathered to the home node which initially hashed the query.

Since the use of a hashing function results in non-relevant material also being accessed, the relevance of each object returned in the search must be determined. This determination of relevance is made by the home node 24 by comparing the degree of similarity between the query 60 and the information or object label 74 returned. In one embodiment the measure of similarity between the query 60 and the object label 74 is a cosine measure and is given by the expression COS (v,w), where the vector v denotes the query 60 and the vector w denotes the content label 74. In one embodiment the N objects with the highest similarity are returned. In another embodiment all objects labels 74 which generate cosine values greater than a predetermined value are considered sufficiently similar to the query 60 to be returned to the user 10 as relevant information.

Once the similarity is determined, the home node 24 orders the OIDs according to their degree of similarity and then returns a list 72 of the most relevant OIDs 70 directly to the user computer 10 or any other computer specified (Step 6), by way of a

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

wide area network 20 or local area network 20 without the intervention of the front end computer 14.

Alternatively, for higher levels of service (level 2 and level 3), the home node 24 passes the most relevant OIDs 70 back to the computer nodes 28, 32 (Step 7) which hold the information or content labels 74 identified by the OIDs 70. These computer nodes 28, 32 then pass the requested information or content labels 74 directly to the user 10 (Step 8).

In more detail, the highest priority task being executed on all the nodes of the search engine 16 is the message handling task shown in Figs. 3, 3a and 3b. Each node waits to receive a message (100) which may be exchanged between the front end node 14 and the home node 24; the home node 24 and the query nodes 26, 28, 30, 32; the home node 24 and the client 10; and the query nodes 26, 28, 30, 32, and the client 10. The message includes a TYPE field which identifies the type of message that is being sent, the source and destination addresses of the nodes, the data to be acted upon and weight information. Upon reception of a message the receiving node determines what type of message has been received by first extracting the message TYPE from the message packet (102).

If the message type received is a PROCESS_PROTOCOL message (104), the message has been sent by the front end node in response to a command from the user 10 and received by a node which is now designated the home node 24. The PROCESS_PROTOCOL message therefore contains a command from the user 10 which is transmitted to the home node 24 by the front end node 14. In one embodiment there are five valid commands (106): INSERT_NEW_CONTENT_LABEL,

WINGATHIN, SCHEGIN,
CIAOSHIN & HAYES
TEL (617) 543-2290
FAX (617) 421-0313

DELETE_CONTENT_LABEL, UPDATE_CONTENT_LABEL, QUERY and GET_CONTENT_LABEL. The home node 24 enqueues (108) an INSERT_TASK, a DELETE_TASK, an UPDATE_TASK or a QUERY_TASK, respectively, in response to the command INSERT_NEW_CONTENT_LABEL, DELETE_CONTENT_LABEL, UPDATE_CONTENT_LABEL, or QUERY contained in the message packet. Once the tasks are enqueued, they are executed as they reach the head of the queue.

In response to the GET_CONTENT_LABEL command, the home node 24 does not enqueue a task, but simply sends a RETRIEVE_CONTENT_LABEL message (109) to a query node, as is described in detail below.

Referring also to Fig. 4, if the command is an INSERT_NEW_CONTENT_LABEL, the home node 24 executes an INSERT_TASK. The INSERT_TASK stores the new content label in secondary storage (110) and then fragments the content label (114). The fragments of the content label are then hashed (120) into a hash value having a node number portion and a hash index value. The home node 24 then sends an INSERT_INDEX_TERM message containing the hash index value to the query node 26 specified by the node number portion of the hashed fragment (124). The home node 24 then determines if there is another fragment (128) and if so, retrieves (132) and hashes the next fragment (120). Once all the hashed fragments have been sent to the query nodes 26, 28 the home node 24 sets a TASK_COUNTER variable (138) and NODE_NUMBER variable to zero (140). The NODE_NUMBER variable is compared to the number of nodes in the network (144) and if the variable is less than the number of nodes on the network, the home node 24 sends a CLOSE_INDEXING message to the node 26 designated by the NODE_NUMBER variable (148) and

increments the NODE_NUMBER variable to the next query node 28 in a list of nodes on the network (152). In another embodiment the home node 24 only increments the NODE_NUMBER variable to the next query node 28 to which at least one INSERT_INDEX_TERM message was sent. The CLOSE_INDEXING message instructs the query nodes 26, 28 that all the hashed index terms have been distributed. If the NODE_NUMBER variable is greater than or equal to the number of nodes in the network, the home node 24 waits for TASK_COUNTER (whose function is described below) to equal the number of nodes in the network (156), and once this occurs, sends a confirmation to the user 10 (158) of the completion of the index insertion task and then exits (160). In another embodiment the home node 24 only waits for the TASK_COUNTER to equal the number of nodes to which at least one INSERT_INDEX_TERM message was sent.

When a query node 26, 28 receives an INSERT_INDEX_TERM message (170) Fig. 3, the query node 26, 28 inserts the hashed index term into its local index table (172) using any of the algorithms known to those skilled in the art and loops to receive the next message (100). When the query node 26, 28 receives a CLOSE_INDEXING message, indicating that all the new index terms have been distributed, the query node 26, 28 replies with a CLOSE_MODIFYING message to the source or home node 24 (182) and again loops for the next message (100).

When the source or home node 24 receives a CLOSE_MODIFYING message (190), the home node 24 locates the task which was enqueued and generated the CLOSE_INDEXING message to which the query node is responding (194) and increments the TASK_COUNTER associated with

that task (198). As shown in Fig. 4, it is upon the incrementing of the TASK_COUNTER that the INSERT_TASK is waiting (156). Once the TASK_COUNTER is incremented (198) the home node 24 loops to receive another message (100).

Referring again to Fig. 3, if the command is a DELETE_CONTENT_LABEL, the home node 24 enqueues and executes a DELETE_TASK. The DELETE_TASK (Fig. 5) first deletes the content label from secondary storage (200). The variables TASK_COUNTER (204) and NODE_NUMBER (208) are then set to zero. The home node 24 then determines if the variable NODE_NUMBER is less than the number of nodes on the network (212) and if so, sends a DELETE_INDEX_TERM message to the node designated by the current NODE_NUMBER (214). The home node 24 then increments the variable NODE_NUMBER (216) and loops. If the variable NODE_NUMBER is greater than or equal to the number of nodes on the network, the home node 24 waits for the variable TASK_COUNTER to equal the number of nodes (218) and then sends a confirmation message to the client 10 that the DELETE_TASK has completed (220) prior to exiting (222).

Referring to Fig. 3a, when the query node 26, 28 receives the DELETE_INDEX_TERM, it enqueues a CANCEL_TASK to remove all the entries in the local database corresponding to the index term to be deleted (234) and loops to receive another message (100). Referring to Fig 6, the CANCEL_TASK begins by reading an index term in the local index (240) and finding the OID which corresponds to that term (244). The query node 26,28 performs a sequential search of all OIDs in the hash table. Sequential scanning of the hash table is used because, in this embodiment, the original content

label being deleted is not known, so it cannot be fragmented and hashed as it is for the INSERT task. In other embodiments, it is possible to use the original content label and is desirable to do so if content labels are frequently updated. A determination is then made as to whether the OID is equal to the OID being deleted (248). If the OID is to be cancelled, the index term for this OID is deleted (250). A determination is then made as to whether there is another OID equal to the OID being deleted (252). If there is another OID equal to the OID being deleted, the index term for this OID is cancelled (250). If there is not another OID corresponding to the OID being deleted, a determination is made as to whether there is another index term (254) and if so, the next term in the index is read (240). If there are no more terms in the local hash table, the query node 26, 28 sends a CLOSE_MODIFYING message (260) to the source of the command or the home node 24 and exits (262).

As described with respect to the INSERT_INDEX_TERM command, (Fig. 3a) when the source or home node 24 receives a CLOSE_MODIFYING message (190), the home node 24 locates the task which was enqueued and which generated the DELETE_INDEX_TERM message to which the query node 26, 28 is responding (194) and increments the TASK_COUNTER associated with that task (198). It is upon this incrementing that the DELETE_TASK is waiting (218) Fig. 5. Once the TASK_COUNTER is incremented (198) the home node 24 loops to receive another message (100).

Referring again to Fig. 3, if the command is a UPDATE_CONTENT_LABEL (106), the home node 24 enqueues and executes an UPDATE_TASK (108). The UPDATE_TASK (Figs. 7 and 7a) is a

combination of an DELETE_TASK followed by an INSERT_TASK, without a confirmation being sent at the end of the DELETE_TASK. The UPDATE_TASK first deletes the content label from secondary storage (600) and then sets the variables TASK_COUNTER (604) and NODE_NUMBER (608) to zero. The home node 24 determines if the variable NODE_NUMBER is less than the number of nodes on the network (612) and if so, sends a DELETE_INDEX_TERM message to the node designated by the current NODE_NUMBER (614), increments the variable NODE_NUMBER (616) and loops. If the variable NODE_NUMBER is greater than or equal to the number of nodes on the network, the home node 24 waits for the variable TASK_COUNTER to equal the number of nodes (618). When TASK_COUNTER is equal to the number of nodes, the task stores the new content label in secondary storage (620) and then fragments the content label (624). As described previously, the fragments of the content label are then hashed (628) into a hash value having a node number portion and a hash index value. The home node 24 then sends an INSERT_INDEX_TERM message containing the hash index value to the query node 26 specified by the node number portion of the hashed fragment (630). The home node 24 then determines if there is another fragment (634) and if so, retrieves (636) and hashes the next fragment (628). Once all the hashed fragments have been sent to the query nodes 26, the home node 24 sets a TASK_COUNTER variable (638) and NODE_NUMBER variable to zero (642). The NODE_NUMBER variable is compared to the number of nodes in the network (644) and if the variable is less than the number of nodes on the network, the home node 24 sends a CLOSE_INDEXING message to the node 26 designated by the

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

NODE_NUMBER variable (648) and increments the NODE_NUMBER variable to the next query node 28 in a list of nodes on the network (652). As discussed with respect to the INSERT_TASK, in another embodiment the home node 24 only increments the NODE_NUMBER variable to the next query node 28 to which the INSERT_INDEX_TERM message was sent. If the NODE_NUMBER variable is greater than or equal to the number of nodes in the network, the home node 24 waits for TASK_COUNTER to equal the number of nodes in the network (656), and once this occurs, sends a confirmation to the user 10 (658) of the completion of the UPDATE_TASK and then exits (660). The response by the query node to the messages sent by the UPDATE_TASK have been described above with respect to the INSERT_TASK and the DELETE_TASK.

Referring to Fig. 3, if the command is a QUERY (106), the home node 24 enqueues and executes a QUERY_TASK (108) as described above. Referring also to Figs. 8 and 8a, the home node 24 fragments the query 60 (270) and hashes the fragment (272) using the same hashing function that was used by the INSERT_TASK as disclosed above. The home node 24 then sends a PROBE message (276) to the node 26 whose address is indicated by the first portion of the hashed fragment. The PROBE message to the query node 26 includes the second portion of the hashed query fragment. The home node 24 then determines whether there is another fragment of the query to process (280). If the is another fragment, that fragment is obtained (270), hashed (272) and transmitted to the node indicated by the first portion of the hashed fragment (276).

If there are no further fragments to process, the home node 24 sets the TASK_COUNTER variable (284) and the NODE_NUMBER variable

(286) to zero, and a determination is made whether the NODE_NUMBER is less than the number of nodes in the network (288). If NODE_NUMBER is less than the number of nodes, a CLOSE_PROBING message is sent (290) to the query node indicated by the NODE_NUMBER variable (to inform the query node that all the query fragments have been sent to the query nodes) and the NODE_NUMBER variable is incremented (292).

Referring again to Fig 3a, when a query node 26 receives a PROBE message, the query node 26 searches its local index for a match to the second portion of the hashed fragment contained in the PROBE message (304). If no matching index term exists (308), the query node 26 simply awaits the next message (100). If a match exists, the OID having the hash value is accessed (310) and the query node 26 sends a GATHER_MATCHING_TERMS message back to the home node 24 (312). The query node 26 then determines if there is another entry in the index matching the hash value in the PROBE message (314). If another entry in the database is matched by the index term, the query node 26 again gets the OID (310) and sends a GATHER_MATCHING_TERMS message back to the home node (310). If no entries are matched by the index term, the query node 26 loops to receive the next message (100).

Similarly, when a query node 26 receives a CLOSE_PROBING message from the home node 24 (320), the query node 26 sends a CLOSE_MATCHING message to the source or home node 24 (324). When the home node 24 receives a CLOSE_MATCHING message from the query node 26 (328), it locates the QUERY task (330) to which the query

WOODCARTER, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 345-2290
FAX (617) 451-0313

node 26 is responding and increments the variable QUERY_COUNTER (334) and loops to receive another message (100).

When the home node 24 receives a GATHER_MATCHING_TERMS message from the query node 26 (340), the home node 24 locates (342) the QUERY task which generated the probe 62 which caused the GATHER_MATCHING_TERMS message to be sent and locates the object entry in the task (344). The home node 24 then adds the weight of the matching term to the total weight of the object (348) and returns to await another message (100).

In one embodiment, the weight of the matching term is computed using a COS function. The query and the index term are treated as vectors with the weight of the matching term determined by the COS function COS(v,w), where v is the query vector and w is the index term vector.

Similarly, the total weight of the object is determined from the COS function according to the following formula:

$$COS(v,w) = \frac{\sum_t v_t w_t}{|v||w|}$$

The sum is over all terms (i.e. both index and query terms) and the product $v_t w_t$ is non-zero only if the term occurs both in the query and in the content label. The length of the vectors v and w are computed by the formulae:

$$|v| = \sqrt{\sum_t (v_t)^2}$$

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

$$|w| = \sqrt{\sum_i (w_i)^2}$$

If v represents the query, then $|v|$ is never actually computed since weights are only considered relative to one another for the same query. The value of $|w|$ is precomputered and what is actually stored are the normalized content label vectors $w/|w|$.

Referring again to Fig. 8, when the QUERY_COUNTER is equal to the number of nodes (350) all the requested data has been received and the object labels are sorted by weight to determine the closeness of the matching of the query 60 and the content labels 74 retrieved (354). The remaining instructions executed by the QUERY task, depend upon the query level. If the query level is one, only the OIDs with the highest weights are returned to the client 10 (360) and the QUERY task exits (362).

If the query level is two, the OIDS with the highest weights are selected and a determination is made as to which objects correspond to those highest weight OIDs (370). The highest weight content labels are selected (372) and the list of highest weight content labels 72 are sent by way of a RETRIEVE_CONTENT_LABEL message to the query node 26 (374) having a node number equal to the information object identifier modulo the number of nodes. That is, the OID modulo the number of nodes can be regarded as another hash function that maps the OID to the node containing the content label. The set of nodes that store content labels in secondary storage need not be the same set as the one that stores the hash table in main memory. The hash function for fragments maps a

WEINGARTEN, SCHURGIN
GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

fragment to the set of matching OIDs. The hash function for OIDs maps an OID to the node containing the corresponding content label. The content label contains information on how to acquire the information objects themselves. If no further objects exist, the QUERY task exits (380).

Referring also to Fig. 3b, when a query node 26 receives a RETRIEVE_CONTENT_LABEL (430), it locates the content label in secondary storage (434) and sends the level 2 query response to the client 10 (436). The query node 26 then loops for another message (100).

Referring again to Fig. 9a, as with level two, if the query level is three, the highest weight OIDs are selected, a determination is made as to which objects correspond to those highest weight OIDs (384) and those objects are selected (388). A FETCH_CONTENT_LABEL message is sent to the query node 26 (390) again corresponding to the information object identifier modulo the number of nodes.

Referring again to Fig. 3b, when the query node 26, receives a FETCH_CONTENT_LABEL message 438, the query node 26 finds the content label in secondary storage 440 and sends a CONTENT_LABEL message to the source of the FETCH_CONTENT_LABEL (442). When the home node 24, receives the CONTENT_LABEL message (446), it locates the QUERY task which generated the FETCH_CONTENT_LABEL message (448), finds the information object entry within the QUERY task (450) and copies the content label to the information object entry (454). The home node 24 then loops to receive another message (100).

Referring again to Fig. 8a, if no further objects exist in the list, the QUERY task waits for all the content labels to be obtained (400). Once all have been obtained, the QUERY task obtains a content label (404) and compares the content label with the query 60 (408). The QUERY task then computes the weight of the label (412). Unlike the weight computed in level 1 and 2 processing, which use fragments of a limited size, the weight computed in level 3 processing considers fragments of any size. In addition, the fragments of the content label that match fragments of the query are marked in the content label. Therefore, if the content label is shown to the user who requested the query, the marked fragments can be distinguished, for example by highlighting, to show the user the reason why the content label was chosen in response to the user's query. Highlighting also serves to focus the user's attention on those portions of the content label that are most likely to be relevant and aid in helping the user formulate subsequent queries. A determination is then made as to whether there are more content labels to process (416) and if so, the next content label is fetched (404). If there are no further content labels, the labels are sorted by weight (418) and the content labels with the highest weights are sent to the client 10 (422). The QUERY task then exits (426).

Referring to Fig. 3, the command DONE is issued to terminate network activity, for example, to permit the software or ontology to be modified. If the message received from the front end node 14 is DONE 508, the activity on the network is terminated and is accompanied by the gathering and printing of activity statistics by

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

the home node 24 (510). When the statistics are printed, the task exits (504).

A graph showing the 95th percentile response time versus throughput (number of queries processed per second) for a four node 550 and an eight node 552 embodiment of system of Fig. 1 is shown in Fig. 9. In this example the database consists of 80,000 information objects indexed by four nodes and 160,000 information objects indexed by eight nodes. The content labels have, on average, 200 index terms, and the queries have, on average, 10 fragments. Each probe produces, on average, four GATHER_MATCHING_TERMS messages at the query node.

It should be noted that tests on a 16-node system have resulted in the same response time graph as that for an 8-node system. Careful monitoring of the system has shown that statistical fluctuations dissipate quickly and the system is very stable as long as the bandwidth of the network is not approached too closely. The network vendor, for the network used in the embodiment disclosed, recommends that the network not be used for transmitting more than 40% of the rated maximum bandwidth. Thus for the embodiment disclosed, this limit is about 500 Kbytes/sec (4Mbits/second). It should be noted, the actual response time for any system is very sensitive to the properties of the database and the distribution of queries. Thus, the throughput achieved by any given system may vary.

Having shown the preferred embodiment, those skilled in the art will realize many variations are possible which will still be within the scope and spirit of the claimed invention. Therefore,

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

it is the intention to limit the invention only as indicated by the scope of the claims.

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

CLAIMS

What is claimed is:

1. A method of accessing data in a distributed database system having a home node and a plurality of query nodes connected by a network, said method comprising the steps of:

fragmenting, by said home node, a query from a user into a plurality of query fragments;

hashing, by said home node, each said query fragment of said plurality of query fragments, said hashed query fragment having a first portion and a second portion;

transmitting, by said home node, each said hashed query fragment of said plurality of query fragments to a respective one of said plurality of query nodes indicated by said first portion of each said hashed query fragment;

using, by said query node, said second portion of said respective hashed query fragment to access data according to a local hash table located on said query node; and

returning, by each said query node accessing data according to said respective hashed query fragment, an object identifier corresponding to said accessed data to said home node.

2. The method of claim 1 further comprising the step of receiving, at said home node, said query from said user, prior to the step of fragmenting said query.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

3. The method of claim 1 further comprising the steps of:

determining, by said home node, a measure of relevance between said accessed data and said query; and

returning, to said user, by said home node, accessed data having a predetermined degree of relevance,

subsequent to the step of returning said object identifier.

4. The method of claim 3 wherein said measure of relevance is determined by a cosine measure.

5. The method of claim 1 wherein said first portion of said hashed query fragment comprises 5 bits and said second portion comprises 32 bits.

6. A method of storing data in a distributed database system having a home node and a plurality of query nodes connected by a network, said method comprising the steps of:

fragmenting, by said home node, data from a user into a plurality of data fragments;

hashing, by said home node, each said data fragment of said plurality of data fragments, said hashed data fragment having a first portion and a second portion;

transmitting, by said home node, each said hashed data fragment of said plurality of data fragments to a respective one of said plurality of query nodes indicated by said first portion of each said hashed data fragment; and

WRINGABTON, SCHUBERT, GAGGESEN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

using, by said query node, said second portion of said respective hashed data fragment to store data according to a local hash table located on said query node.

7. The method of claim 6 further comprising the step of receiving, at said home node, said data from said user, prior to the step of fragmenting said data.

8. A distributed database system comprising:

a home node; and

a plurality of query nodes;

said home node and said plurality of query nodes connected by a network,

said home node, upon receiving a query from a user, fragments said query into a plurality of query fragments, hashes each said query fragment of said plurality of query fragments into a hashed query fragment having a first portion and a second portion, and transmits each said hashed query fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed query fragment,

said query node, uses said second portion of said hashed query fragment to access data according to a local hash table located on said query node and returns, an object identifier corresponding to said accessed data to said home node.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

9. The distributed database system of claim 8 wherein said home node determines a measure of relevance between said accessed data and said query and returns to said user accessed data having a predetermined degree of relevance.

10. The method of claim 9 wherein said home node measures relevance using a cosine measure.

11. The method of claim 8 wherein said first portion of said hashed query fragment comprises 5 bits and said second portion comprises 32 bits.

12. A distributed database system comprising:

a home node; and

a plurality of query nodes;

said home node and said plurality of query nodes connected by a network.

said home node, upon receiving data from a user, fragments said data into a plurality of data fragments, hashes each said data fragment of said plurality of data fragments into a hashed data fragment having a first portion and a second portion, and transmits each said hashed data fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed data fragment,

said query node, uses said second portion of said hashed data fragment to store data according to a local hash table located on said query node.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

13. A distributed database system comprising a home node and a plurality of query nodes connected by a network,

said home node, upon receiving a command from a user, enqueueing a predetermined task in response to said command,

a query task enqueued, in response to a query command from said user, fragmenting a query contained in said query command into a plurality of query fragments, hashing each said query fragment of said plurality of query fragments into a hashed query fragment having a first portion and a second portion, and transmitting a query message containing each said hashed query fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed query fragment,

said query node, upon receipt of said query message, using said second portion of said hashed query fragment to access data according to a local hash table located on said query node and transmitting a message returning an object identifier corresponding to said accessed data to said home node.

14. The method of claim 13 wherein said query message requests predetermined data from said query node in response to a query level contained in said query command from said user.

15. The method of claim 14 wherein there are three query levels.

16. The method of claim 14 wherein said query node returns a content label in response to a predetermined query level.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

17. A distributed database system comprising a home node and a plurality of query nodes connected by a network,

said home node, upon receiving a command from a user, enqueueing a predetermined task in response to said command,

an insert task enqueued, in response to an insert command from said user, fragmenting data contained in said insert command into a plurality of data fragments, hashing each said data fragment of said plurality of data fragments into a hashed data fragment having a first portion and a second portion, and transmitting an insert message containing each said hashed data fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed data fragment,

said query node, upon receipt of said insert message, using said second portion of said hashed data fragment to store data according to a local hash table located on said query node.

NU-360XX
TAT/dmk
51140.WP

WEEMARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

Attorney
Docket No.: MU-360XX

DECLARATION AND POWER OF ATTORNEY

As a below-named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

the specification of which (check one):

[X] is attached hereto. [ ] was filed _____ as Serial No. _____; amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations §1.56(a).

I hereby claim foreign priority benefits under Title 35 USC 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which Priority is claimed:

Prior Foreign Application(s)		Date Filed	Priority Claimed	
(Number)	(Country)	(Day/Month/Year)	[ ] Yes	[ ] No
(Number)	(Country)	(Day/Month/Year)	[ ] Yes	[ ] No
(Number)	(Country)	(Day/Month/Year)	[ ] Yes	[ ] No

I hereby claim the benefit under Title 35 USC 120 of any United States application(s) listed below and insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35 USC 112. I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial No.)	(Filing Date)	(Patented/pending/abandoned)
(Application Serial No.)	(Filing Date)	(Patented/pending/abandoned)
(Application Serial No.)	(Filing Date)	(Patented/pending/abandoned)

10/92 FORM 11

Sheet 2 of 2

Attorney
Docket No.: NU-360XX

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) to prosecute this application and transact all business connected therewith in the Patent and Trademark Office, and to file with the USRO any International Application based thereon.

Stanley M. Schurgin, Reg. No. 20,979
Charles L. Gagnebin III, Reg. No. 25,467
Paul J. Hayes, Reg. No. 28,307
Victor B. Lebovici, Reg. No. 30,864
Eugene A. Feher, Reg. No. 33,171
Beverly E. Hjorth, Reg. No. 32,033

Address all correspondence to:

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
Ten Post Office Square
Boston, Massachusetts 02109
Telephone: (617) 542-2290
Telecopier: (617) 451-0313

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor: Kenneth P. Baclawski		
City of Residence: Waltham	State or Country: Massachusetts MA	Country of Citizenship: U.S.A.
Post Office Address: 35 Fairmont Avenue	City: Waltham	State or Country: Massachusetts Zip Code: 02154
Signature: (Please sign and date in permanent ink.) X [signature] ✓		Date signed: X 10/4/94 ✓

Full Name of Second Joint Inventor:		
City of Residence	State or Country	Country of Citizenship
Post Office Address	City	State or Country Zip Code
Signature: (Please sign and date in permanent ink.) X		Date signed: X

53634.WP

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANT:	Kenneth P. Baclawski	ATTORNEY DOCKET NO.:	NU-360XX
SERIAL NO.:		EXAMINER:	
FILED:	HEREWITH	GROUP NO.:	
PATENT NO.:		ISSUED:	
ENTITLED:	DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD		

VERIFIED STATEMENT AS SMALL ENTITY

Honorable Commissioner of Patents and Trademarks
Washington, D.C. 20231

Sir:

THE UNDERSIGNED DECLARE(S):

Exclusive rights in the above-identified invention reside in the "small entity(ies)" defined and named below, and "small entity" fees are appropriate. Qualification as a small entity is based upon the appropriately checked statements below:

[ ] INDEPENDENT INVENTOR(S)

The below-signing independent inventor(s) has (have) not assigned, granted, conveyed or licensed, and is (are) under no obligation under contract or law to assign, grant, convey or license any rights in the invention to any person who could not likewise be classified as an independent inventor under 37 CFR 1.9(c) if that person had made the invention, or to any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e).

[ ] SMALL BUSINESS CONCERN

The below-identified small business concern qualifies as a small business as defined in 13 CFR 121.3-18, and reproduced in 37 CFR 1.9(d), for purposes of paying reduced fees under Section 41(a) and (b) of Title 35, in that the number of employees, including those of its affiliates, which does not exceed 500 persons, and it has not assigned, granted, conveyed or licensed, and is under no obligation under contract or law to assign, grant, convey or license, any rights in the invention to any person who could not be classified as an independent inventor under 37 CFR 1.9(c) if that person had made the invention, or to any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e).

Concerns are affiliates of each other when, either directly or indirectly, one concern controls or has the power to control the other, or a third party controls or has the power to control both. The number of employees of the business concern is the average over the fiscal year of the persons employed during each of the pay periods of the fiscal year. Employees are those persons employed on a full-time, part-time or temporary basis during the previous fiscal year of the concern.

Attorney
Docket No.: NU-360XX

[x] **NONPROFIT ORGANIZATION** (Check additional applicable box.)

The below-identified nonprofit organization qualifies as a small entity under 37 CFR 1.9(e) in that it constitutes:

1. [X] a university or other institution of higher education located in any country; or

2. [ ] an organization of the type described in Section 501(c)(3) of the Internal Revenue Code of 1954 (26 USC 501(c)(3)) and exempt from taxation under Section 501(a) of the Internal Revenue Code (26 USC 501(a)); or

3. [ ] any nonprofit scientific or educational organization qualified under a nonprofit organization statute of a state of the United States (35 USC 201(i)); or

4. [ ] any nonprofit organization located in a foreign country which would qualify as a nonprofit organization under paragraphs (e)(2) or (3) of Rule 1.9 if it were located in the United States.

The undersigned acknowledge(s) the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate (37 CFR 1.28(b)).

The below-signing individual(s) hereby declare(s) that (he, she, they) are authorized to execute this statement on behalf of the small entity; that all statements made herein of (his, her, their) own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

Name of Small Entity: (Independent Inventor/Small Business/Nonprofit) NORTHEASTERN UNIVERSITY	
Address of Small Entity: (Street, City, State or Country, Zip Code) 360 HUNTINGTON AVENUE, BOSTON, MA 02115	
Name of Person Signing: (Small Business/Nonprofit) RICHARD J. MCNEIL, JR.	
Title of Person Signing: (Small Business/Nonprofit) ASSOCIATE DIRECTOR	
Signature: (Please sign and date in permanent ink.) X	Date Signed: X October 6, 1994

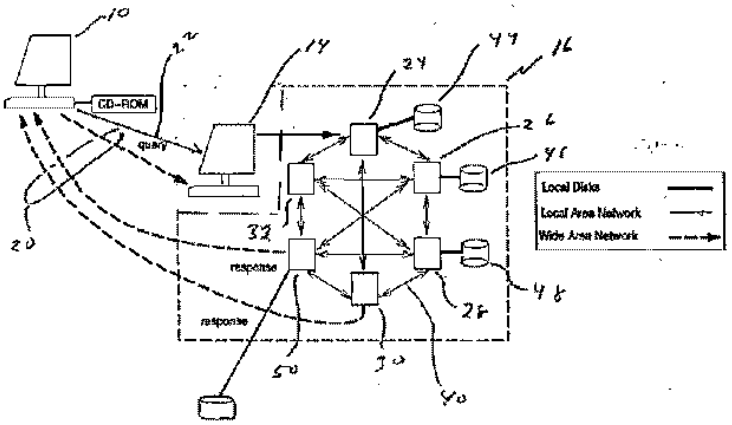


Fig. 1

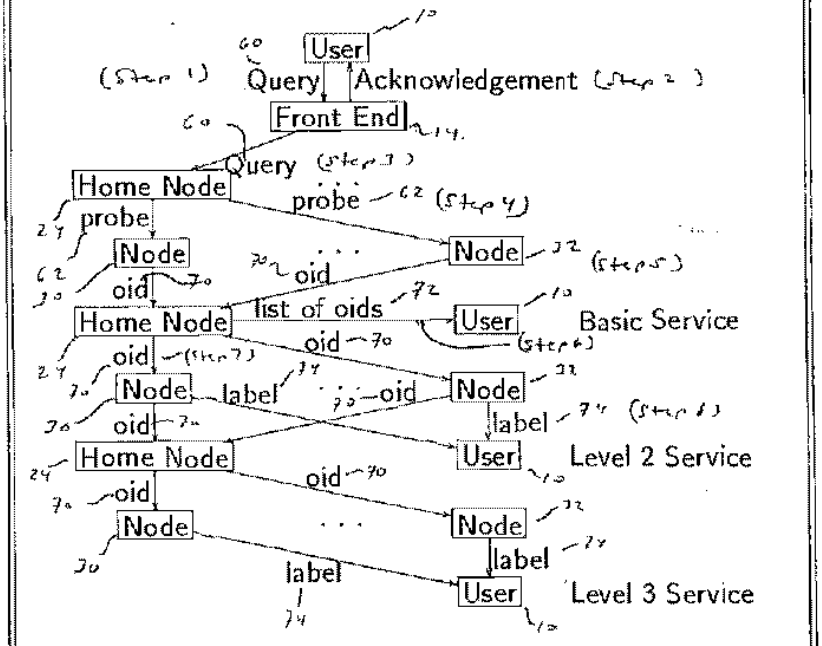

User
Query
Acknowledgement
Front End
Query
Home Node
probe
Node
oid
list of oids
User
Basic Service
Level 2 Service
Level 3 Service
label

Fig 2

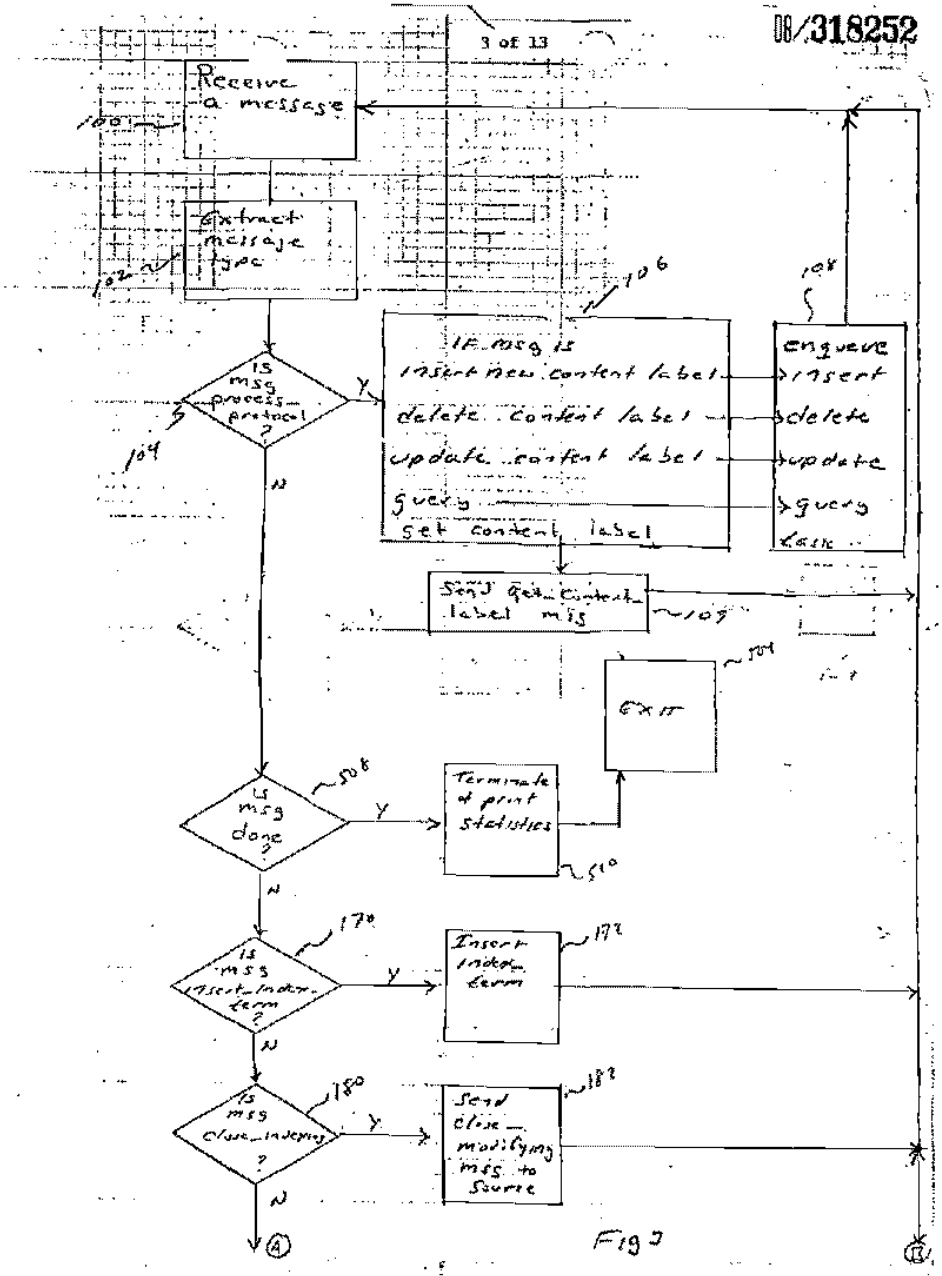

Receive a message
100
Extract message type
102
Is msg process_protocol ?
104
Y
N
IF msg is
insert new content label
delete content label
update content label
query
set content label
106
engine
insert
delete
update
query
Send Set_content_label msg
107
108
EXIT
104
Is msg done ?
506
Terminate & print statistics
510
Is msg insert_index_term ?
170
Insert index term
172
Is msg close_index ?
180
Send close_ modifying msg to Source
182
A
B

Fig 3

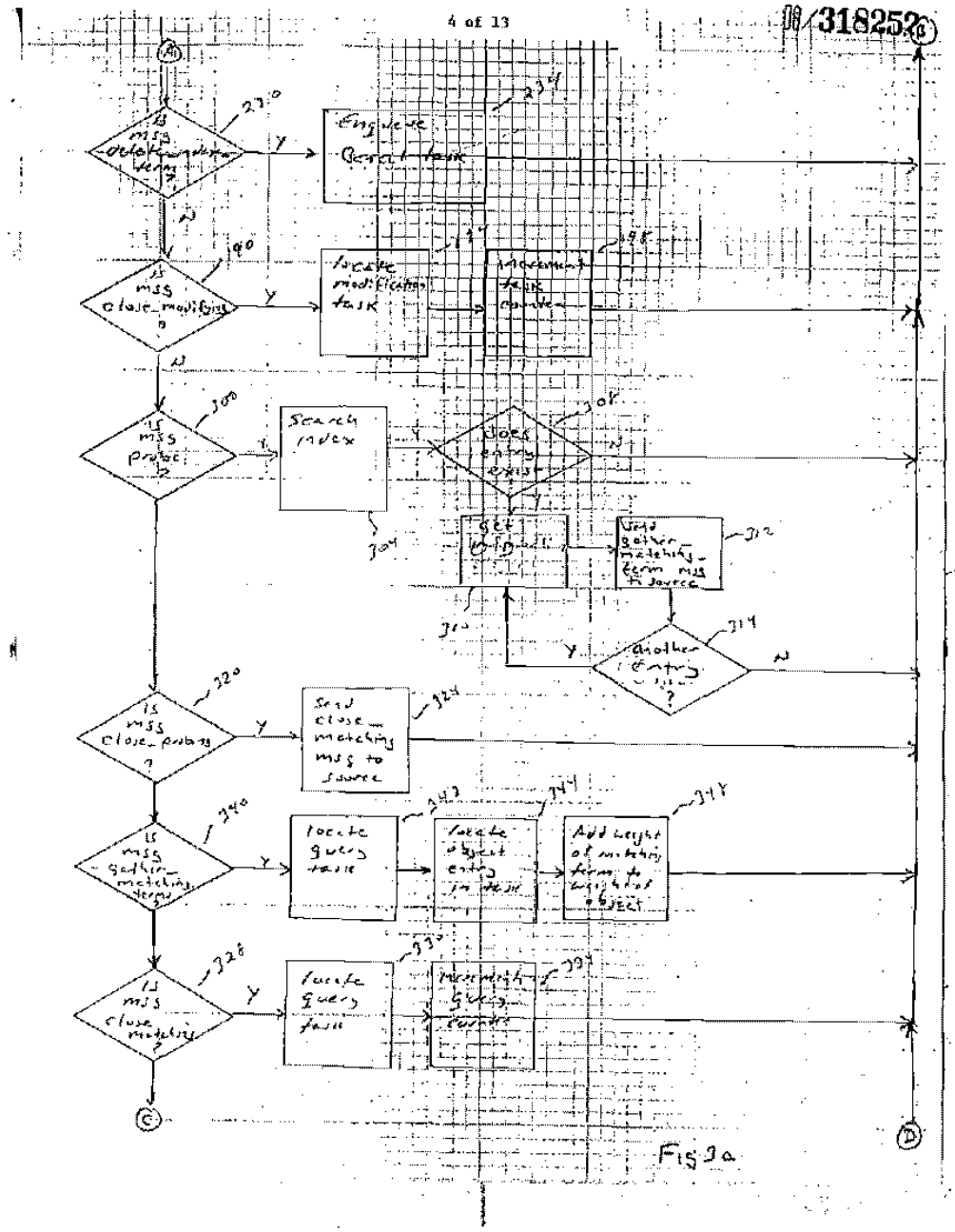

A
Is msg close_modifying?
Y
N
Is msg close_modifying?
Y
Locate modification task
Movement task created
N
Is msg probe?
Y
Search index
Does entry exist?
or
N
Y
Get object
Send gather_matching_term msg to source
Another entry exists?
Y
N
Is msg close_probing?
Y
Send close_matching msg to source
Is msg gather_matching_term?
Y
Locate query task
Locate object entry in table
Add weight of matching term to weight of object
Is msg close_matching?
Y
Locate query task
Increment query counter
C
D

Fig 3a

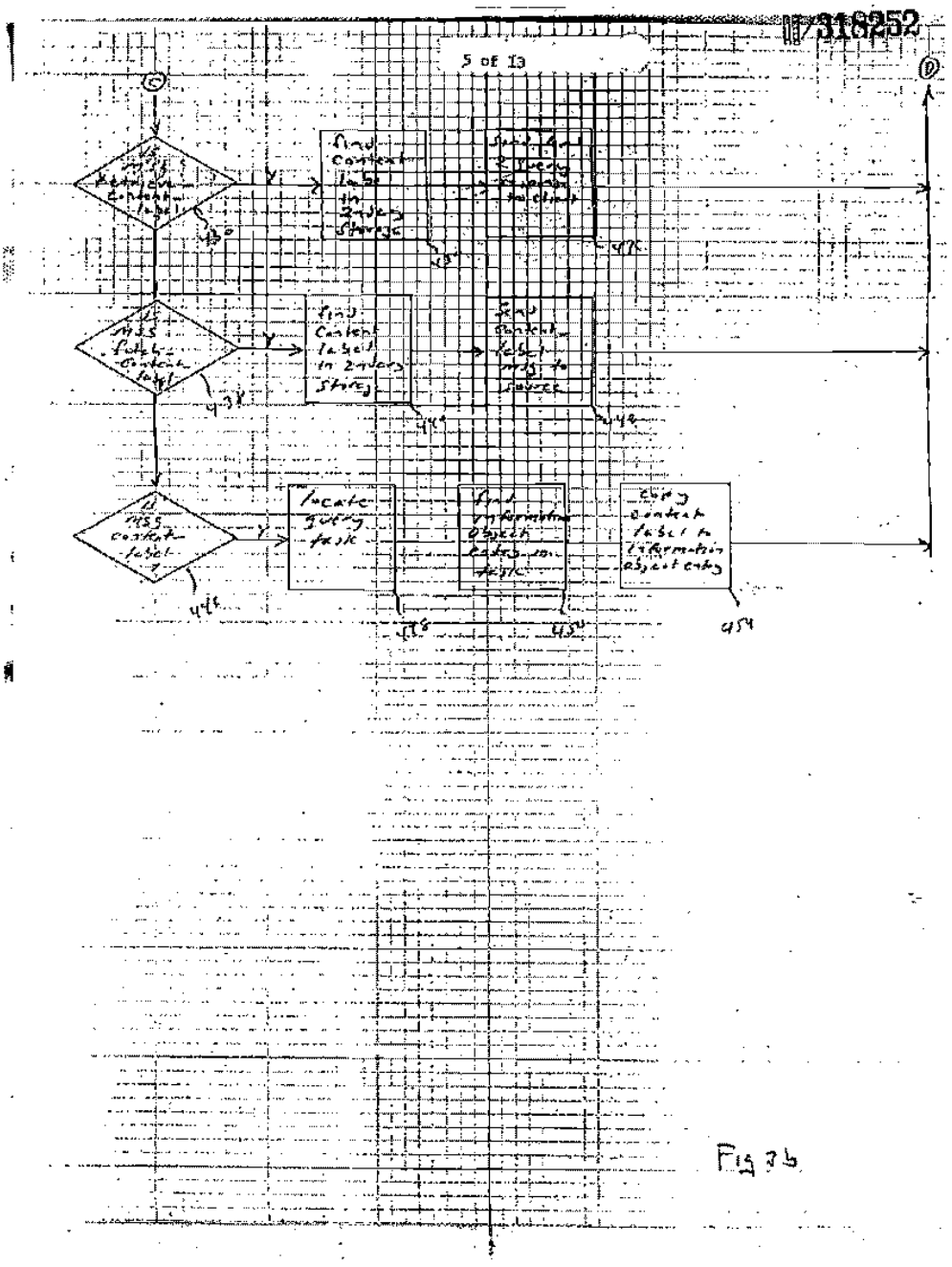


Fig 3b

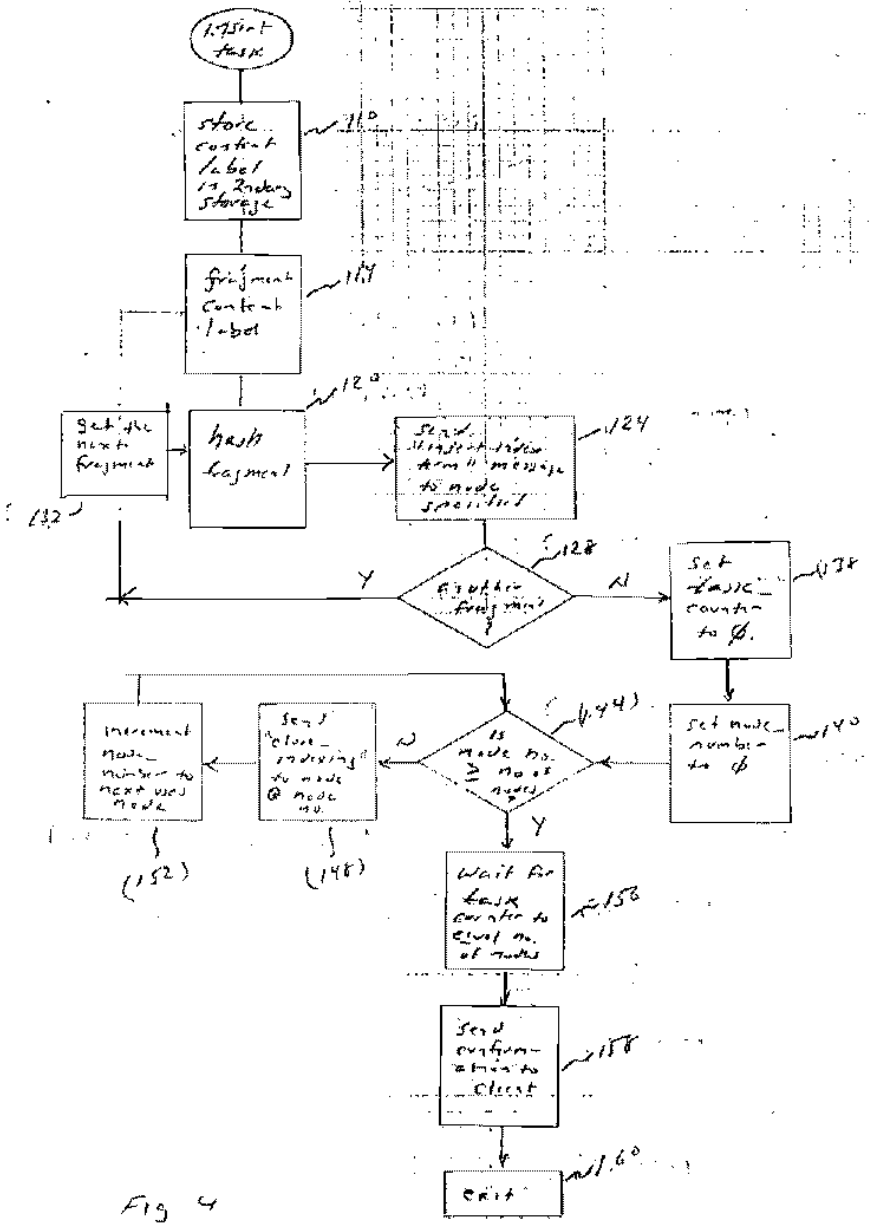


Fig 4

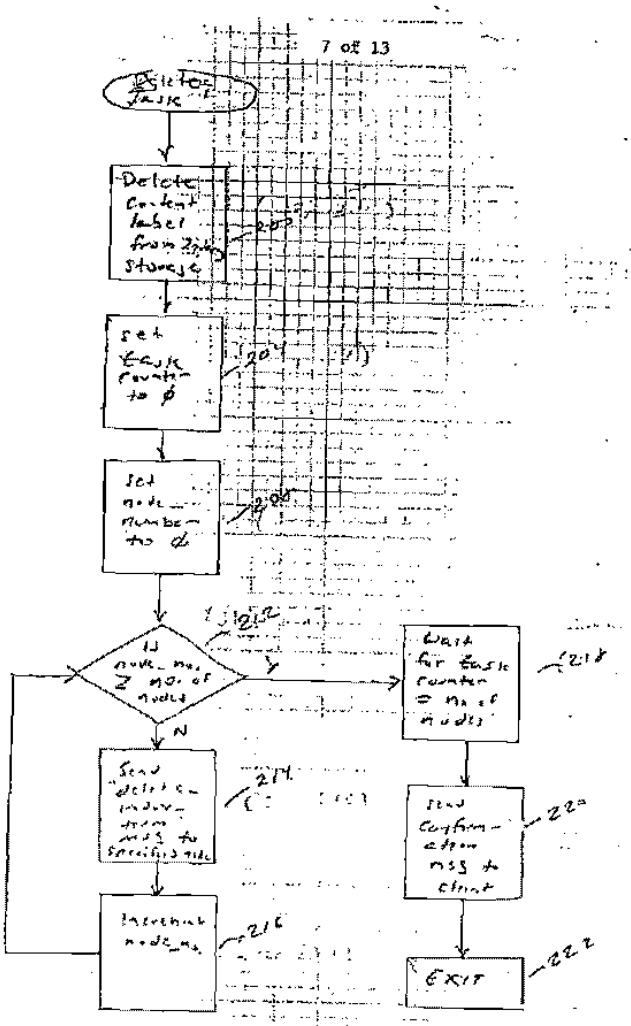


Fig 5

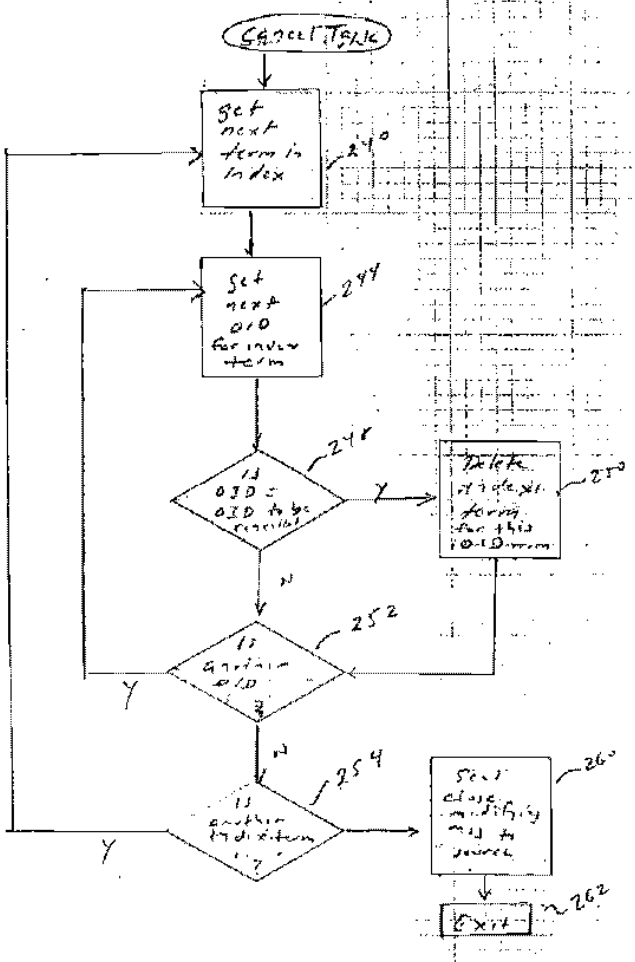

CANCEL TALK
Get next term in index
240
Set next OID for index term
244
Is OID = OID to be cancelled
246
Y
Delete index term for this ID-term
250
N
Is another OID
252
Y
N
Is another index item
254
Y
Send close notification msg to source
260
Exit
262

Fig. 6.

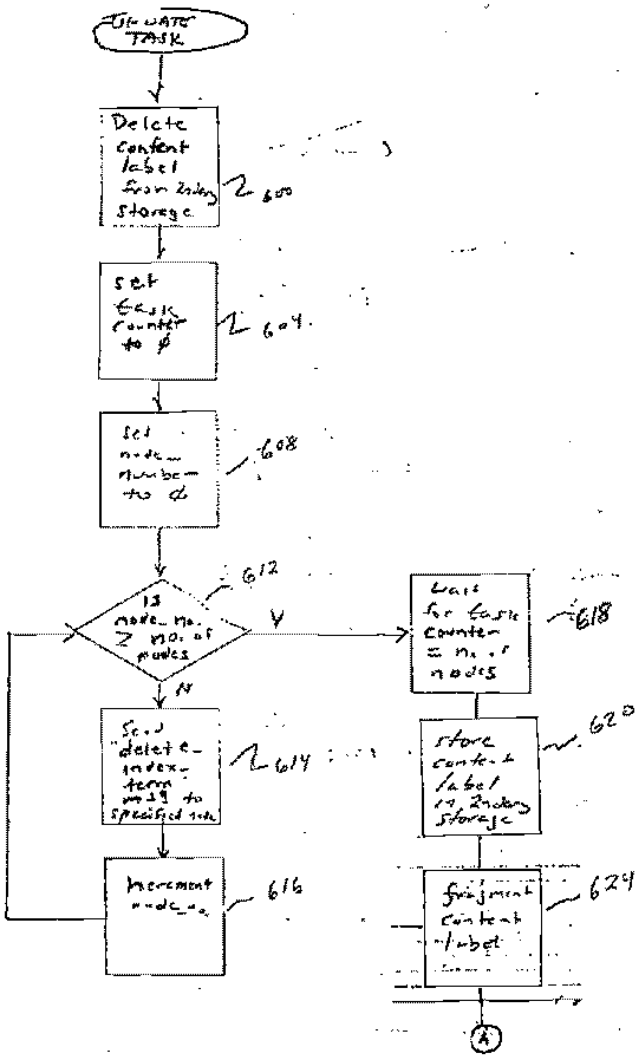

UPDATE TASK
Delete content label from index storage
600
Set task counter to 0
604
Set node_number to 0
608
612
Is node_no. ≥ no. of nodes
Y
N
Wait for task counter = no. of nodes
618
Send "delete_index_term msg" to specified node
614
Store content label in index storage
620
Increment node_no.
616
Fwd segment content label
624
A

Fig 6

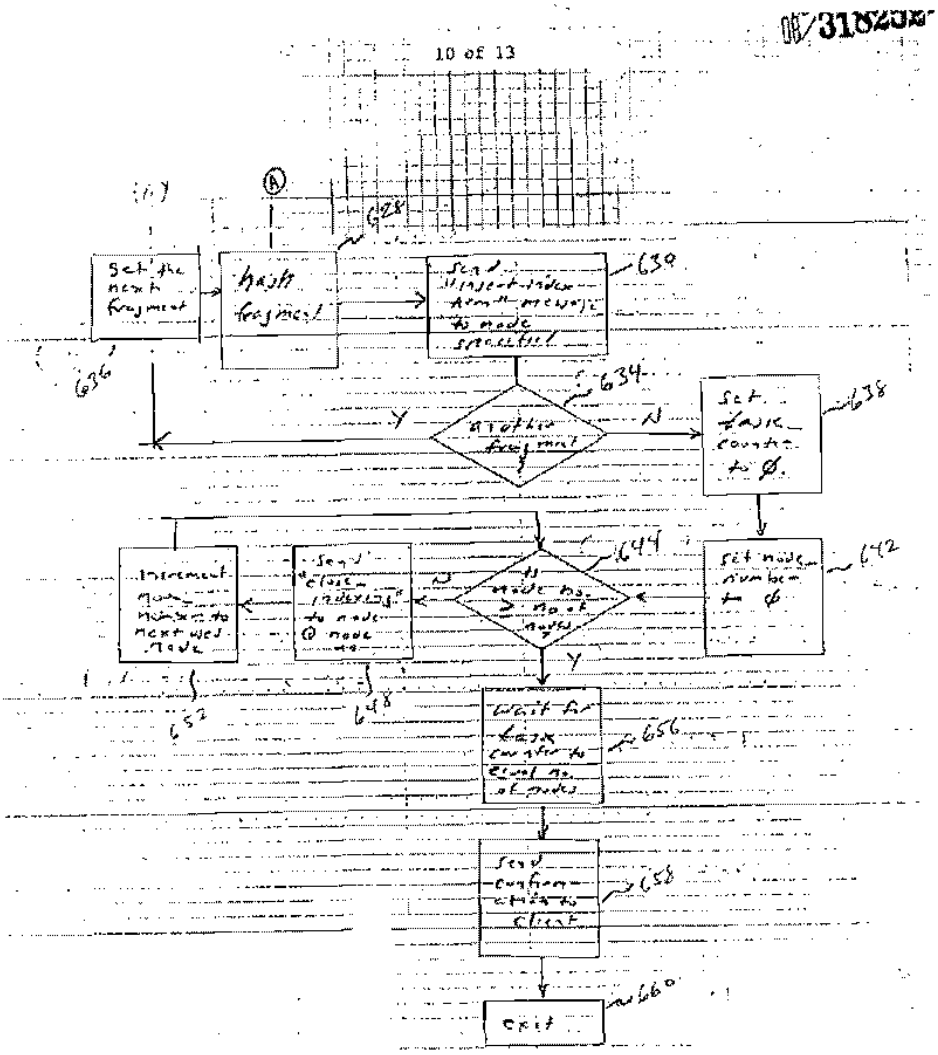
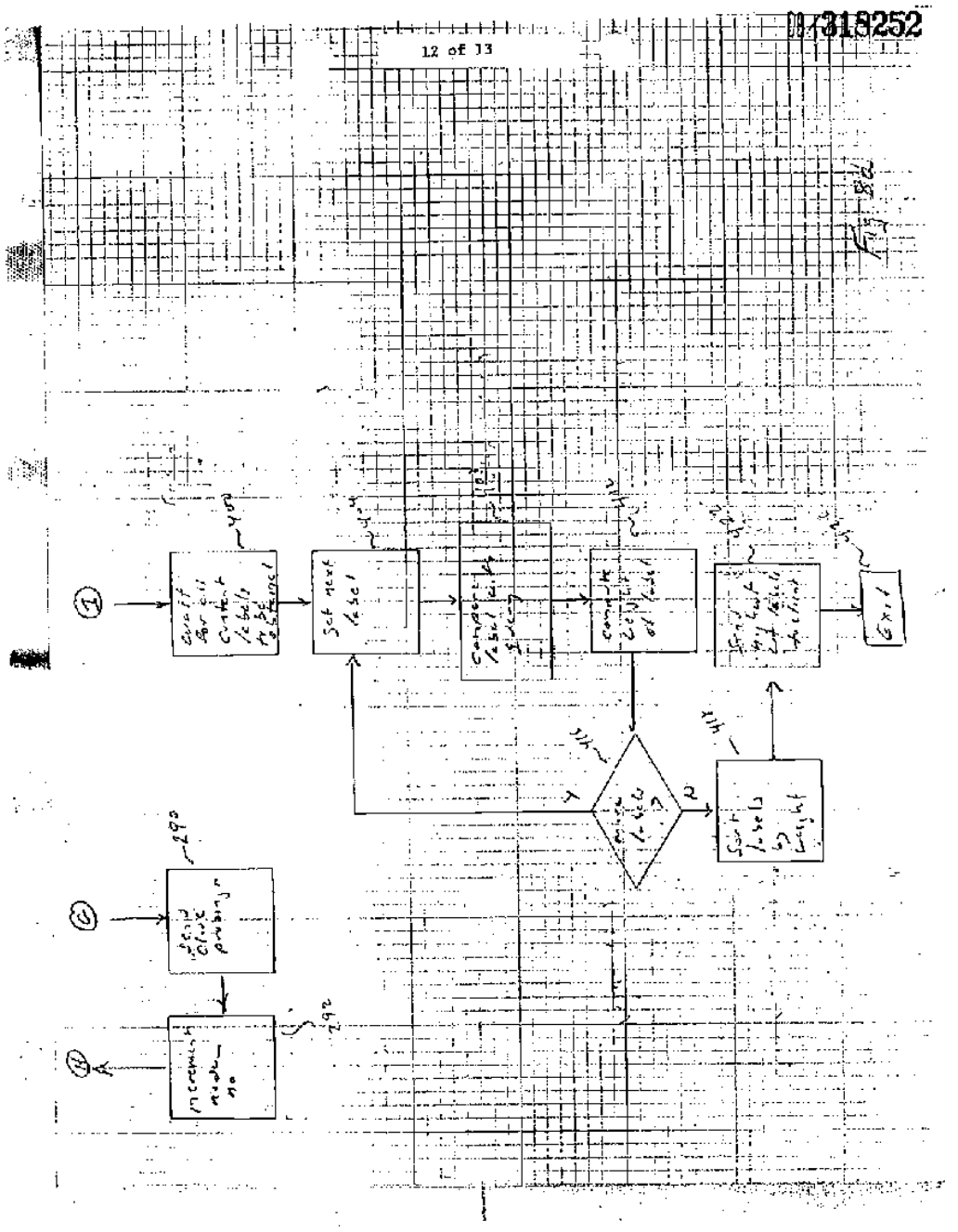

Set the next fragment
last fragment?
Y
N
Set basic counter to 0
Increment node number to next node
node no. > no. of nodes
Y
set node number to 0
wait for
exit
628
630
634
638
642
644
646
648
652
656
658
660

Fig 7c



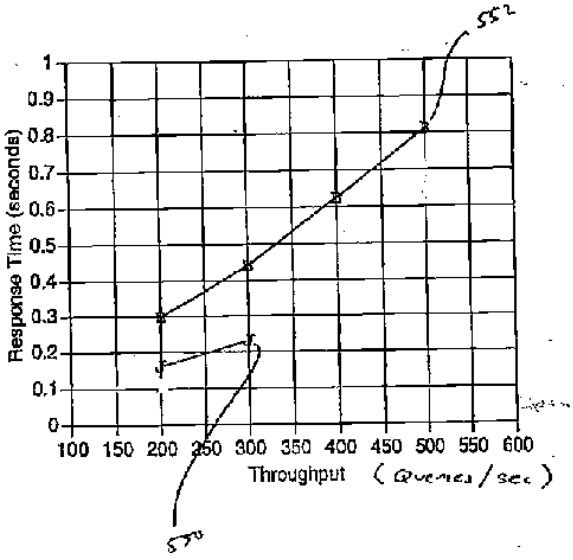

552
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Response Time (seconds)
100 150 200 250 300 350 400 450 500 550 600
Throughput (Queries/sec)
550

Fig 9

PATENT APPLICATION FEE DETERMINATION RECORD

Effective October 1, 1994

Application or Docket Number: 318252

CLAIMS AS FILED - PART I

(Column 1) FOR	NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY RATE	SMALL ENTITY FEE	OR	OTHER THAN SMALL ENTITY RATE	OTHER THAN SMALL ENTITY FEE
BASIC FEE				365.00	OR		730.00
TOTAL CLAIMS	17 minus 20 =	*	x$11=		OR	x$22=	
INDEPENDENT CLAIMS	6 minus 3 =	* 3	x38=	114	OR	x76=	
MULTIPLE DEPENDENT CLAIM PRESENT			+120=		OR	+240=	
			TOTAL	479	OR	TOTAL	

* If the difference in column 1 is less than zero, enter "0" in column 2

CLAIMS AS AMENDED - PART II

	(Column 1)	CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE	SMALL ENTITY ADDITIONAL FEE	OR	OTHER THAN SMALL ENTITY RATE	OTHER THAN SMALL ENTITY ADDITIONAL FEE
AMENDMENT A	Total	* 17	Minus	** 17	=	x$11=		OR	x$22=	
	Independent	* 6	Minus	*** 6	=	x38=		OR	x76=	
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+120=		OR	+240=	
						TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	
AMENDMENT B	Total	* 17	Minus	** 20	=	x$11=		OR	x$22=	
	Independent	* 6	Minus	*** 6	=	x38=		OR	~~x76=~~	
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+120=		OR	~~+240=~~	
						TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	0
AMENDMENT C	Total	*	Minus	**	=	x$11=		OR	x$22=	
	Independent	*	Minus	***	=	x38=		OR	x76=	
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+120=		OR	+240=	
						TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

FORM PTO-875 (Rev. 10/94) — Patent and Trademark Office, U.S. DEPARTMENT OF COMMERCE

JAR0002610

Form PTO 1190 (REV 9/94)

U.S. DEPARTMENT OF COMMERCE
Patent and Trademark Office

PACE DATA ENTRY CODING SHEET

1ST EXAMINER	DATE
Bates	11-3
2ND EXAMINER	DATE

APPLICATION NUMBER	TYPE APPL	FILING DATE (MONTH DAY YEAR)	SPECIAL HANDLING	GROUP ART UNIT	CLASS	SHEETS OF DRAWING
08/318252	1	10 05 94	0	3717	395	13

TOTAL CLAIMS	INDEPENDENT CLAIMS	SMALL ENTITY?	FILING FEE	FOREIGN LICENSE	ATTORNEY DOCKET NUMBER
11	6	1	479	Y	NU360XX

CONTINUITY DATA

CONT CODE	STATUS CODE	PARENT APPLICATION SERIAL NUMBER	PCT APPLICATION SERIAL NUMBER	PARENT PATENT NUMBER	PARENT FILING DATE (MONTH DAY YEAR)
			PCT/ /		
			PCT/ /		
			PCT/ /		
			PCT/ /		
			PCT/ /		

PCT/FOREIGN APPLICATION DATA

FOREIGN PRIORITY CLAIMED	COUNTRY CODE	PCT/FOREIGN APPLICATION SERIAL NUMBER	FOREIGN FILING DATE (MONTH DAY YEAR)

JAR0002611

BAR CODE LABEL

U.S. PATENT APPLICATION

SERIAL NUMBER	FILING DATE	CLASS	GROUP ART UNIT
08/318,252	10/05/94	395	2317

APPLICANT

KENNETH P. BACLAWSKI, WALTHAM, MA.

CONTINUING DATA********
VERIFIED

FOREIGN/PCT APPLICATIONS********
VERIFIED

FOREIGN FILING LICENSE GRANTED 11/04/94 ***** SMALL ENTITY *****

STATE OR COUNTRY	SHEETS DRAWING	TOTAL CLAIMS	INDEPENDENT CLAIMS	FILING FEE RECEIVED	ATTORNEY DOCKET NO.
MA	13	17	6	5479.00	NU360XX

ADDRESS

WEINGARTEN SCHURGIN GAGNEBIN & HAYES
TEN POST OFFICE SQUARE
BOSTON MA 02109

TITLE

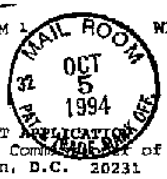
DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

This is to certify that annexed hereto is a true copy from the records of the United States Patent and Trademark Office of the application which is identified above.

By authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

Date Certifying Officer

PTO/92 FORM 1



WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
Ten Post Office Square
Boston, Massachusetts 02109
Telephone: (617) 542-2290
Telecopier: (617) 451-0313

08/318252

BOX PATENT APPLICATION
Honorable Commissioner of Patents and Trademarks
Washington, D.C. 20231

Date: October 5, 1994

Attorney
Docket No.: NU-350XX

Sir:

Transmitted herewith for filing is the patent application of:

Inventor: Kenneth P. Baclawski

Entitled: DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

Enclosed are:

[X] 13 sheets of informal drawings (one set)
[X] an Assignment of the invention to: NORTHEASTERN UNIVERSITY (w/original signature)
[ ] a Certified copy of a _____ application
[X] a Verified statement re small entity status (§1.9 and §1.27) (w/original signature)
[ ] Information Disclosure Statement
[X] Other: A Declaration and Power of Attorney (w/original signature)

[ ] Continuation-in-part application of application Serial No._____, filed _____

[X] Thomas A. Turano is hereby appointed Associate Attorney by:
Registration No.: 35,722

Attorney of Record: Stanley M. Schurgin
Registration No.: 20,979

CLAIMS FILED:	MINUS BASE:	EXTRA CLAIMS:	RATE:	BASIC FEE:
				$730.00
Independent	6 - 3	= 3	x $76.00 =	228.00
Total	17 - 20	= 0	x $22.00 =	0
[ ] Multiple Dependent Claims (1st presentation)			+ $240.00 =	0
			SUBTOTAL	958.00
Small Entity filing, divide by 2. (Note: verified statement must be attached per §1.9, §1.27, §1.28.)				479.00
			TOTAL FILING FEE	479.00

[X] The filing fee has been calculated above; a check in the amount of $479.00 is enclosed.
[ ] The filing fee will be submitted at a later date.
[X] The Commissioner is hereby authorized to charge payment of any additional filing fees under §1.16 associated with this communication or credit any overpayment to Deposit Account No. 23-0804.

SUBMIT IN TRIPLICATE
53787.WP

Attorney of Record: Thomas A. Turano
Registration No.: 35,722

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application	:	Kenneth P. Baclawski
Filed	:	HEREWITH
	:	DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD
Attorney's Docket	:	NU-360XX

* *

Express Mail Mailing Number HB157242811US
Date of Deposit - October 5, 1994

I hereby certify that the following items are being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and as addressed to BOX PATENT APPLICATION, Commissioner of Patents and Trademarks, Washington, D.C. 20231:

U.S. Patent application of Kenneth P. Baclawski, entitled DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD, consisting of:

Specification includes: PP 1 through 22 of Abstract through Detailed Description; PP 23 through 28 of claims 1-17.

Drawings as follows (one copy informal): First sheet of Fig. 1; Second sheet of Fig. 2; Third Sheet of Fig. 3; Fourth sheet of Fig. 3a; Fifth sheet of Fig. 3b; Sixth sheet of Fig. 4; Seventh sheet of Fig. 5; Eighth sheet of Fig. 6; Ninth sheet of Fig. 7; Tenth sheet of Fig. 7a; Eleventh sheet of Fig. 8; Twelfth sheet of Fig. 8a; and Thirteenth sheet of Fig. 9;

and including Declaration and Power of Attorney (w/original signature), together with a check in the amount of $ 479.00 to cover the filing fee thereof and a cover letter in triplicate.

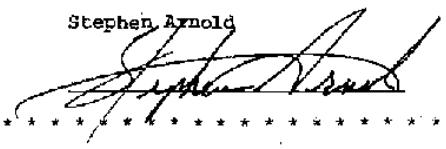
An Assignment of the invention and application for recording of Kenneth P. Baclawski to NORTHEASTERN UNIVERSITY comprising 2/pages (w/original signature) with a cover letter signed by Thomas A. Turano;

a check in the amount of $40.00 to cover the Assignment recording fee; and

A Verified Statement As Small Entity (w/original signature).

The above items are deposited with signatures and dated by the filing attorney as appropriate.

Stephen Arnold



* *

TAT/dmk
53796.WP

10/92 FORM 1

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
Ten Post Office Square
Boston, Massachusetts 02109
Telephone: (617) 542-2290
Telecopier: (617) 451-0313

08/318252

BOX PATENT APPLICATION
Honorable Commissioner of Patents and Trademarks
Washington, D.C.

Date: October 5, 1994

Attorney
Docket No.: NU-360XX

Sir:

Transmitted herewith for filing is the patent application of:

Inventor: Kenneth P. Baclawski

Entitled: DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

Enclosed are:

[X] 13 sheets of informal drawings (one set)
[X] an Assignment of the invention to: NORTHEASTERN UNIVERSITY (w/original signature)
[ ] a Certified copy of a _____ application
[X] a Verified statement re small entity status (§1.9 and §1.27) (w/original signature)
[ ] Information Disclosure Statement
[X] Other: A Declaration and Power of Attorney (w/original signature)

[ ] Continuation-in-part application of application Serial No._____, filed _____

[X] Thomas A. Turano is hereby appointed Associate Attorney by:
Registration No.: 35,722

Attorney of Record: Stanley M. Schurgin
Registration No.: 20,979

CLAIMS FILED:	MINUS BASE:	EXTRA CLAIMS:	RATE:	BASIC FEE:
				$730.00
Independent	6 - 3	= 3	x $76.00 =	228.00
Total	17 - 20	= 0	x $22.00 =	0
[ ] Multiple Dependent Claims (1st presentation)			+ $240.00 =	0
			SUBTOTAL FILING FEE	958.00
Small Entity filing, divide by 2. [Note: verified statement must be attached per §1.9, §1.27, §1.28.)				479.00
			TOTAL FILING FEE	479.00

[X] The filing fee has been calculated above: a check in the amount of $479.00 is enclosed.
[ ] The filing fee will be submitted at a later date.
[X] The Commissioner is hereby authorized to charge payment of any additional filing fees under §1.16 associated with this communication or credit any overpayment to Deposit Account No. 23-0804.

SUBMIT IN TRIPLICATE
53787.WP

Attorney of Record: Thomas A. Turano
Registration No.: 35,722

PTO/SB FORM

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
Ten Post Office Square
Boston, Massachusetts 02109
Telephone: (617) 542-2290
Telecopier: (617) 451-0313

OB/318252

BOX PATENT APPLICATION
Honorable Commissioner of Patents and Trademarks
Washington, D.C. 20231

Date: October 5, 1994

Attorney
Docket No.: NU-360XX

Sir:

Transcitted herewith for filing is the patent application of:

Inventor: Kenneth P. Baclawski

Entitled: DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

Enclosed are:

[X] 13 sheets of informal drawings (one set)
[X] an Assignment of the invention to: NORTHEASTERN UNIVERSITY (w/original signature)
[ ] a Certified copy of a _____ application
[X] a Verified statement re small entity status (§1.9 and §1.27) (w/original signature)
[ ] Information Disclosure Statement
[X] Other: A Declaration and Power of Attorney (w/original signature)

[ ] Continuation-in-part application of application Serial No._____, filed _____

[X] Thomas A. Turano is hereby appointed Associate Attorney by:
Registration No.: 35,722

Attorney of Record: Stanley M. Schurgin
Registration No.: 20,979

CLAIMS FILED:	MINUS BASE:	EXTRA CLAIMS:	RATE:	BASIC FEE:
				$730.00
Independent	6 - 3	= 3	x $76.00 =	228.00
Total	17 - 20	= 0	x $22.00 =	0
[ ] Multiple Dependent Claims (1st presentation)			+ $240.00 =	0
			SUBTOTAL FILING FEE	958.00
Small Entity filing, divide by 2. (Note: verified statement must be attached per §1.9, §1.27, §1.28.)				479.00
			TOTAL FILING FEE	479.00

[X] The filing fee has been calculated above; a check in the amount of $479.00 is enclosed.
[ ] The filing fee will be submitted at a later date.
[X] The Commissioner is hereby authorized to charge payment of any additional filing fees under §1.16 associated with this communication or credit any overpayment to Deposit Account No. 23-0804.

Attorney of Record: Thomas A. Turano
Registration No.: 35,722

SUBMIT IN TRIPLICATE
53787.WP

PATENT

2301
#2
01-9-95

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application : Kenneth P. Baclawski
Serial No. : 08/318,252
Filed : October 5, 1994
For : DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD
Attorney's Docket : NU-360XX

Group Art Unit: 2307 2317

* *

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Honorable Commissioner of Patents and Trademarks, Washington, D.C. 20231 on 17 Nov 94.

By _____
Thomas A. Turano
Registration No. 35,722
Attorney for Applicant

* *

INFORMATION DISCLOSURE STATEMENT

Honorable Commissioner of Patents and Trademarks
Washington, D.C. 20231

Sir:

It is desired to cite for the record in this application the enclosed articles and U.S. patents listed on the attached copy of PTO Form #1449. The paragraph(s) marked below are applicable to this Information Disclosure Statement.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

Serial No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2317

[X] (1) The enclosed Information Disclosure Statement is being filed within three months of the filing date or within three months of the entry of the national stage of the above identified application. Accordingly, applicant(s) believe that no fee or certification is required.

[ ] (1a) Applicant(s) believe the enclosed Information Disclosure Statement is entitled to the benefit of 37 CFR §1.97(b)(3). Accordingly, applicant(s) believe that no fee or certification is required.

[ ] (1b) Pursuant to 37 CFR §1.97(c), the enclosed Information Disclosure Statement is being filed before the mailing date of a final action or a notice of allowance and is accompanied by:
[ ] a certification under 37 CFR §1.97(e); [ ] the fee set forth in §1.17(p).

PETITION UNDER 37 CFR §1.97(d)

[ ] (2) Pursuant to 37 CFR §1.97(d), applicant(s) hereby petition the Commissioner to consider the attached Information Disclosure Statement. Applicant(s) state that the issue fee has not been paid and that a certification under 37 CFR §1.97(e) is provided herein, along with the petition fee of $130.00 required under 37 CFR §1.17(i)(1).

WEINGARTEN, SCHURGIN,
GAGNEBIN & LEVAY
TEL (617) 542-2290
FAX (617) 451-0313

Serial No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2317

CERTIFICATION UNDER 37 CFR §1.97(e)(1)

[ ] (3) The undersigned hereby certifies that each item of information contained in the attached Information Disclosure Statement was cited in a communication from a foreign patent office in a counterpart foreign application mailed not more than three months prior to the filing of this statement.

CERTIFICATION UNDER 37 CFR §1.97(e)(2)

[ ] (4) The undersigned hereby certifies that no item of information contained in the attached Information Disclosure Statement was cited in a communication from a foreign patent office in a counterpart foreign application or, to the knowledge of the undersigned, after making reasonable inquiry, was known to any individual having a duty of disclosure as set forth in 37 CFR §1.56(c) more than three months prior to the filing of this statement.

[ ] (5) The information disclosure fee of $200.00 required by 37 CFR §1.17(p) is believed to be due and is enclosed herewith.

The filing of this Information Disclosure Statement is not a representation by the undersigned as to personal knowledge of the contents of every word or phrase of the material enclosed or that reliance on other suitably trained professionals has not been made.

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

Serial No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2317

If a search report of a searching agency is enclosed identifying the nature of the relevance of each document, such a designation is deemed to satisfy Rule 98(a)(3) even if in a foreign language, since the few terms of relevance therein are deemed of universal cognizance. However, applicant does not necessarily adopt the position reflected by that report.

The Commissioner is hereby authorized to charge payment of any additional fees associated with this communication or credit any overpayment to Deposit Account No. 23-0804. Triplicate copies of this letter are enclosed.

Respectfully submitted,

KENNETH P. BACLAWSKI

By _____
Thomas A. Turano
Registration No. 35,722
Attorney for Applicant

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
Ten Post Office Square
Boston, Massachusetts 02109

Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: 17 Nov 94

TAT/dmk
S6167.WP

Enclosures

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

Form PTO-1449 (REV. 8-83) — U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE

ATTY. ET NO.	SERIAL NO.
NU-36000	08/318,252
APPLICANT: Kenneth P. Baclawski	
FILING DATE: October 5, 1994	GROUP: 2317

INFORMATION DISCLOSURE CITATION

(Use several sheets if necessary)

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE*
CL		4 8 1 1 1 9 9	3/7/89	Keuchler et al.	364	200	
CL		5 0 0 6 9 7 8	4/9/91	Heches	364	200	
CL		5 3 0 9 3 5 9	5/3/94	Katz et al.	364	419.19	
CL		5 2 6 5 2 0 7	11/23/93	Zak et al.	395	200	

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	TRANSLATION YES	TRANSLATION NO

OTHER DOCUMENTS *(Including Author, Title, Date, Pertinent Pages, Etc.)*

CL	*Baclawski, K., "High-Performance Indexing and Retrieval for Object-Oriented Databases", College of Computer Science, Northeastern University, April 1994.*
CL	*Baclawski et al., "High Performance, Distributed Information Retrieval", College of Computer Science, Tecnical Report NU-CCS-94-05, Northeastern University, February 25, 1994.*
CL	*Salton, G., "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer", Cornell University, Chapter 10, December 1988.*
CL	*Salton et al., "Automatic Structuring and Retrieval of Large Text Files", Communications of the ACM, Vol. 37, No.2, February 1994.*
CL	*Baclawski et al., "A distributed approach to high-performace information retrieval", Northeastern University, March, 1994.*
CL	*Baclawski et al., "A unified approach to high-performance, vector-based information retrieval", Northeastern University, March 21, 1994.*

EXAMINER: Cheryl Lewis — DATE CONSIDERED: 2/6/96

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Form PTO-1449 (REV. 8-83)

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE

INFORMATION DISCLOSURE CITATION

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.
NU-560XX	08/318,252
APPLICANT: Kenneth P. Baclawski	
FILING DATE: October 5, 1994	GROUP: 2317

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE*

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	TRANSLATION YES	TRANSLATION NO

OTHER DOCUMENTS *(Including Author, Title, Date, Pertinent Pages, Etc.)*

CL		*Baclawski et al., "KEYNET: An architecture and protocol for high-performance semantically rich information retrieval", Northeastern University, April 29, 1994.*
CL		*Baclawski et al., "An abstract for semantically rich information retrieval", Northeastern University, March 31, 1994.*
CL		*Baclawski et al., "KEYNET: Fast indexing for semantically rich information retrieval", Northeastern University, Technical Report NU-CCS-94-06, December 7, 1993.*

EXAMINER: Charles D. Knowles

DATE CONSIDERED: 3/15/96

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

WSGR FORM 12
(8/95)

THESE PUBS ARE FOUND AS IS IN THE FILE.

SORRY FOR THE INCONVENIENCE

High Performance, Distributed Information Retrieval

by

Kenneth Baclawski and J. Elliott Smith

College of Compute: Science
Technical Report NU-CCS-94-05

Northeastern University
Boston

keywords.

In addition, the keynet framework allows for sequences of concepts linked by relationships and expressed in natural language using phrases, clauses, sentences and paragraphs. No current systems use such a capability, so there is no evidence that it would have a significant impact on retrieval effectiveness. Nevertheless, the KEYNET system establishes that there is no technological or user-interface barrier to such semantically rich IR methods.

Some examples of information objects that would be well-suited to retrieval using the KEYNET technique include:

- Scientific research papers. This was the original motivation for the development of the technique. We are developing natural language processing tools for extracting keynets directly from the document[BFH+93].
- Scientific data files. Such files must be accompanied by (or include) a description of the contents. If such a description is structured, then it forms a keynet.
- Satellite images. The geographic coordinates, time taken, etc. can be incorporated into a keynet.
- Videotapes. The director, producer, stats, etc., along with a structured description of the content, forms a keynet.

The KEYNET technique is designed to be used in a highly distributed environment. It is assumed that the information objects themselves are widely distributed. At the KEYNET site itself, the keynets are kept on disks and distributed among the nodes of a local-area network. The index to the keynets is kept in the main memories of the nodes of the LAN. Processing of queries as well as insertion of new keynets is done using distributed algorithms.

Each keynet contains information about locating and acquiring the actual information object. The KEYNET system is only concerned with finding information objects. Acquiring (and paying for) objects is an independent issue.

The user's computer is responsible for presentation (user-interface) services. To accomplish this the user must have a copy of the ontology. Using current technology, this would be kept on a CD-ROM. We have developed a prototype interface of this type[BF93].

Queries and responses are sent over the network. The prototype achieves response times that are so fast that access to the local CD-ROM drive would generally be slower than access to the search engine. The prototype system uses the datagram protocol (UDP) of the Internet TCP/IP protocol family.

At the KEYNET site, an interface computer is responsible for relaying query requests to one of the search engine computers. The search engine itself is a collection of processors (or more precisely server processes) joined by a high-speed LAN. A query is itself a keynet. Queries are answered by fragmenting them into terms that are matched against similar fragments obtained from the information objects. A scatter/gather algorithm is employed to distribute the query terms and collect the matching objects. Relevance is measured using standard vector methods, specifically the cosine measure[Sal89].

High-Performance, Distributed Information Retrieval

Kenneth Baclawski* and J. Elliott Smith
Northeastern University
College of Computer Science
Boston, Massachusetts 02115
{kenb,esmith}@ccs.neu.edu

February 25, 1994

We propose an information retrieval (IR) system called KEYNET which is a high-performance, distributed *search engine* for locating *information objects in a large subject-specific* corpus. The system could handle up to a few million information objects with performance at a level of hundreds of queries per second (with current workstation technology).[1] We have *developed a prototype keynet system for testing the validity of the basic ideas and also to* get preliminary performance results.

A KEYNET system requires the development of a subject-specific ontology that is understandable to a literate practitioner of the field. A keynet ontology represents knowledge using a directed graph of conceptual categories and relationships between them. The Unified Medical Language System (UMLS) developed by the National Library of Medicine is an example of such an ontology[HL93, LHM93].

Each information object must be annotated with a small directed graph (called a *keynet*) that indicates what portion of the ontology relates to the content of the object. Keynets are semantically intermediate between keywords and semantic networks[Lev92]. Keynets provide an overall framework that generalizes many commonly used mechanisms for information retrieval, such as: subject classification schemes, keywords, document abstracts, reviews, content labels for non-textual information objects, properties such as author or date of publication, ranges of text strings such as "wild card" match strings, and ranges of quantities. The KEYNET system allows a uniform treatment of these disparate techniques in a system that permits a great deal of flexibility compared to traditional database and information retrieval systems. For example, one could combine *all of the above mechanisms in a single* system, and one can easily add new features to the ontology, such as new attributes and

*This material is based upon work supported by the National Science Foundation Grant No. IRI-9117030.

[1] By comparison, the LEXIS legal information retrieval system of Mead Data Central, answers 250,000 queries per day, an average of 3 queries per second, although the peak load is much higher.

keywords.

In addition, the keynet framework allows for sequences of concepts linked by relationships and expressed in natural language using phrases, clauses, sentences and paragraphs. No current systems use such a capability, so there is no evidence that it would have a significant impact on retrieval effectiveness. Nevertheless, the KEYNET system establishes that there is no technological or user-interface barrier to such semantically rich IR methods.

Some examples of information objects that would be well-suited to retrieval using the KEYNET technique include:

- Scientific research papers. This was the original motivation for the development of the technique. We are developing natural language processing tools for extracting keynets directly from the document[BFH+93].
- Scientific data files. Such files must be accompanied by (or include) a description of the contents. If such a description is structured, then it forms a keynet.
- Satellite images. The geographic coordinates, time taken, etc. can be incorporated into a keynet.
- *Videotapes. The director, producer, stars, etc., along with a structured description of* the content, forms a keynet.

The KEYNET technique is designed to be used in a highly distributed environment. It is assumed that the information objects themselves are widely distributed. At the KEYNET site itself, the keynets are kept on disks and distributed among the nodes of a local-area network. The index to the keynets is kept in the main memories of the nodes of the LAN. Processing of queries as well as insertion of new keynets is done using distributed algorithms.

Each keynet contains information about locating and acquiring the actual information object. The KEYNET system is only concerned with finding information objects. Acquiring (and paying for) objects is an independent issue.

The user's computer is responsible for presentation (user-interface) services. To accomplish this the user must have a copy of the ontology. Using current technology, this would be kept on a CD-ROM. We have developed a prototype interface of this type[BF93].

Queries and responses are sent over the network. The prototype achieves response times that are so fast that access to the local CD-ROM drive would generally be slower than access to the search engine. The prototype system uses the datagram protocol (UDP) of the Internet TCP/IP protocol family.

At the KEYNET site, an interface computer is responsible for relaying query requests to one of the search engine computers. The search engine itself is a collection of processors (or more precisely server processes) joined by a high-speed LAN. A query is itself a keynet. Queries are answered by fragmenting them into terms that are matched against similar fragments obtained from the information objects. A scatter/gather algorithm is employed to distribute the query terms and collect the matching objects. Relevance is measured using standard vector methods, specifically the cosine measure[Sal89].

Responses are sent directly to the requester from the search engine processor that collects the search result. The prototype differs from the proposed architecture only in that it randomly generates the keynet repository as well as queries sent to it.

The prototype runs on a network of up to 8 sparcstations connected by a twisted-pair network. The network is not dedicated to our research project. Among other results, we found that it is feasible to implement a high-performance search engine that "borrows" underutilized resources on a network of workstations.

The prototype is fully distributed, using a pure message-passing communication mechanism. All messages are one-way: no process ever waits for a reply to a message. The memory model is local, i.e., a "shared nothing" system. The individual nodes are implemented as servers. Specifically, they are implemented as connectionless, multi-threaded, interrupt-driven, stateless servers. Threads explicitly yield control and are never preempted, but they can be interrupted. Each server is responsible for a fixed amount of memory, chosen to be small enough for page faulting to be unusual. The indexing uses a hash algorithm using direct addressing with secondary hashing in the event of a collision. Collisions did not have an impact on performance even when the hash tables were 90% full. Each document keynet was generated randomly and had 200 index terms. This corresponds roughly to an document abstract that is around 100 to 150 words long, or equivalently, to a content label with 200 attributes. Queries had 10 terms each. In one run, the throughput for an 8 node network indexing 80,000 documents was 900 queries/sec, with a median response time of 2.3 seconds. At a load of 400 queries per second, the median response time was 0.3 seconds, and more than 95% of the queries were answered in less than 0.6 seconds.

References

[BF93] K. Baclawski and N. Fridman. M&M-Query: Database support for the annotation and retrieval of biological research articles. In *Proc. ACM SIGMOD Conference*, 1993. Submitted.

[BFH+93] K. Baclawski, R. Futrelle, C. Hafner, M. Peschitelli, N. Fridman, B. Li, and C. Zou. Data/knowledge bases for biological papers and techniques. In *Proc. Sympos. Adv. Data Management for the Scientist and Engineer*, pages 23–28, 1993.

[HL93] Betsy L. Humphreys and Donald A.B. Lindberg. The umls project: making the conceptual connection between users and the information they need. *Bulletin of the Medical Library Association*, 81(2):170, Apr 1 1993.

[Lev92] Robert Levinson. Pattern associativity and the retrieval of semantic networks. *Computers and Mathematics with Applications*, 23(6-9):573–600, 1992.

[LHM93] D.A.B. Lindberg, B.L. Humphreys, and A.T. McCray. The unified medical language system. *Methods of information in medicine*, 32(4):281, Aug 1 1993.

[Sal89] Gerard Salton. *Automatic Text Processing*. Addison-Wesley, Reading, MA, 1989.

High-Performance Indexing and Retrieval for Object-Oriented Databases

Kenneth Baclawski

College of Computer Science
Northeastern University
Boston, Massachusetts 02115

The Scientific Databases Project
Data/Knowledge bases for Biological Papers and Techniques

KEYNET:

Prof. Dan Simovici, J. Elliott Smith and Jagadeesh Venugopal.

OntologyBuilder:

Natalya Fridman and Prof. Carole Hafner.

Object-Oriented Natural Language Processing Tools:

Prof. Robert Futrelle, Prof. Carole Hafner, Dr. Maurice J. Pescitelli, Shobana Sampath, Yumiko Sekiya and Xiaolan Zhang.

This material is based upon work supported by the National Science Foundation under Grant No. IRI-9117030.

Typical News Article

Newsgroups: sdb.tw.phone,sdb.news.gov.usa
Distribution: sdb.usa
Subject: New Phone Numbers
Keywords: General financial/business news
Copyright: 1994 by The SDB Press, R
Date: Fri, 1 Apr 94 19:04:02 DBT
ACategory: financial
Slugword: Animal-Phones
Priority: regular
ANPA: Wc: 112/5; Id: Q92532; Src: sdb;
Sel: -----; Adate: 04-01-N/A
Codes: SDB-1313

WASHINGTON (DC) -- The Federal Communications Commission could take the first step toward a new breed of mobile telephones -- phone numbers assigned to an animal, not a person -- by the end of the year.

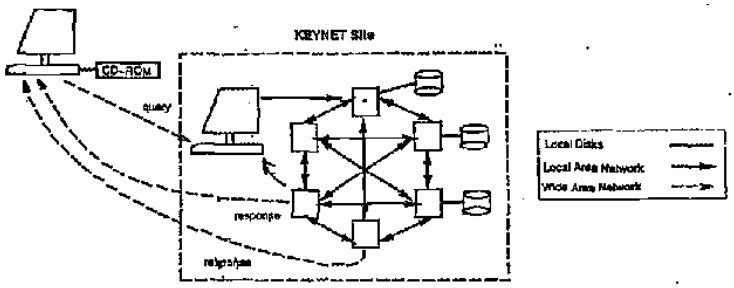
That's when the FCC is likely to begin requiring that all dogs be issued their own phone numbers, Francis Barkington, chief of the FCC's Animal Control Office, said in an interview Friday.

The timing has been unclear and a source of speculation within the pet communications industry.

The FCC is re-thinking a complex plan, adopted last week, which could eventually mandate that every mammal have access to the Internet.

Socks Clinton, the White House spokescreature, remarked "On the Internet nobody thinks I'm a dog."

Locations of Data in KEYNET



Data	Location
Information Objects	Scattered throughout the network
User Interface and Ontology	User/Searcher machine
Content labels	Distributed over local disks at KEYNET site
Index	Distributed hash table stored in KEYNET server main memories

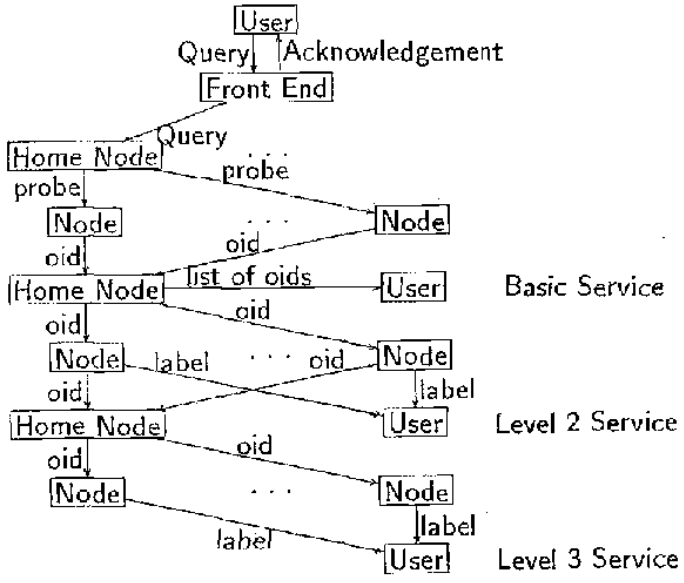
Introduction

- Immense amounts of scientific (and other) information are becoming available electronically.
- Efficient methods for finding information in these new "information superhighways" are essential.
- The KEYNET system is an architecture and algorithm for high-performance information retrieval based on the vector space model.
- The KEYNET model is designed for retrieval from a corpus of information objects in a single subject area. A knowledge model or *ontology* for the subject area is necessary.
- Information objects must be annotated with content labels.
- Both content labels and queries are expressed using the language of the ontology. The model can support queries using many commonly used search mechanisms.
- The KEYNET model can also support semantically rich retrieval mechanisms, such as semantic networks.
- The prototype system achieves a throughput of 500 queries per second with a response time of less than one second on an 8-node network of workstations.

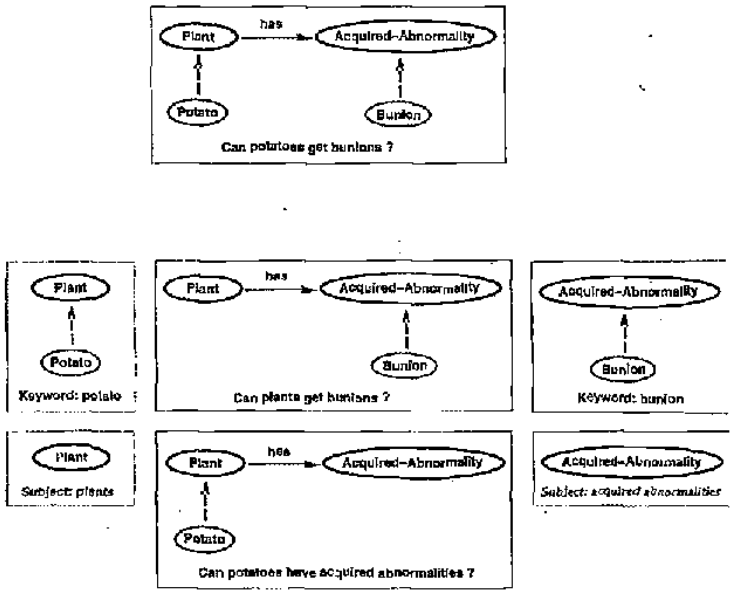
Examples of Corpora and Ontologies

Short news articles	Attribute-value pairs
Videos	Role-person pairs and synopsis
Satellite Images	Attribute-value pairs and Attribute-value range pairs
Scientific data files	Attribute-value and Attribute-value range pairs
Derived scientific data	Process metadata
Molecular biology papers	Materials & Methods
Medical papers	Unified Medical Language System
Scientific research papers	Subject-specific ontology
Genetic mapping data	Probes as index terms
Federal Information Infrastructure	Government Information Locator Service

The KEYNET Distributed IR Algorithm



Fragments of a Query



Glossaries

A keynet ontology can be thought of as a language for expressing concepts in a specific subject area. A keynet is a statement conforming to this language. In particular, one can use keynets to *define* or *explain* lexical terms. Such an explanation is called a *gloss*.

A *gloss* of a term t is a keynet G with a distinguished term g called the *genus proximus* such that all the other terms of G are properties of g or of a property of g (recursively). The properties of g are the *specific difference* of t within g.

Theorem. For any term t in a keynet K conforming to an ontology O, and any gloss G of t, one can substitute G for t in K to obtain a structure $K\lfloor_t G$ which is a keynet conforming to the ontology O.

Keynets and Ontologies

The structure and behavior of keynets are defined by the *Ontology* consisting of:

- A directed graph called the *schema*
  - Vertices represent *conceptual categories*, *types*, *parts of speech* or *attributes*.
  - Links of the schema relate conceptual categories. For example, *ISA*, *part of*, *elaboration of* and *cause of*.
- A *lexicon*. Also called a *thesaurus*. Elements can be *lexical terms*, *keywords* or *concepts*. A lexical term is an instance of at least one conceptual category.

A keynet *conforming* to a keynet ontology consists of:

- A *directed graph*
  - Vertices are copies of vertices in the ontology. One vertex in the ontology can appear more than once in a keynet.
  - Edges are copies of edges in the ontology. The keynet must faithfully reflect the corresponding structure in the ontology.
- A set of copies of lexical terms

Fragments

A *fragment* of a keynet is a connected subset of the keynet.

In other words, a fragment consists of a choice of vertices, edges and lexical terms from a keynet such that the set of vertices and edges so chosen forms a connected directed graph.

The main result for fragments is the fact that the number of fragments grows linearly with the size of the keynet from which they are taken.

Theorem. Let K be a keynet having v vertices and e edges. If the degree of a vertex of K is at most d, then there are at most $2v + 4e + 8e(d - 1)$ fragments of K having at most 2 edges.

Example of a Medical Content Label

Focus groups carry out etiocholanolone, which measures electron transport, neural conduction and denosine-N(6)-methylpropylthioether-N-pyridoxamine, which interacts with riboflavin glucoside, protein-serine-threonine kinase p44(mpk), and potassium triphosphate. Etiocholanolone also affects the ion pump, and measures potash, which interacts with glutathione transferases. Glutathione transferases affect sperm motility.

Etiocholanolone can be used to assess the effect of hydrogen fluoride, alkaline DNase, oxidative phosphorylation and 8-fluorobenzo(a)pyrene, which interacts with acid DNase and alcohol dehydrogenase. In addition, etiocholanolone can be used to assess the effect of 3,4-DAMPe, which interacts with chromium sodium oxide (CrNa2O4) and affects endocytosis and the citric acid cycle.

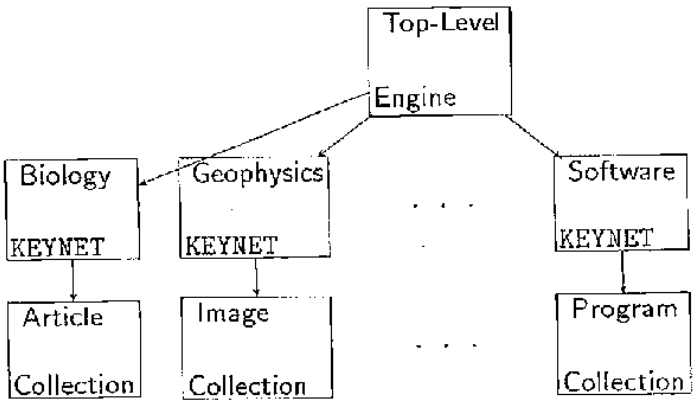
Thiophosphate affects receptor-mediated signal transduction, which produces mannitol dehydrogenase. Mannitol dehydrogenase interacts with 6-azido-FAD and complicates hemagglutination.

Receptor aggregation is a process of neural transmission that produces glypin and is affected by BPFBD, spectrophotometry NEC and dicentrine.

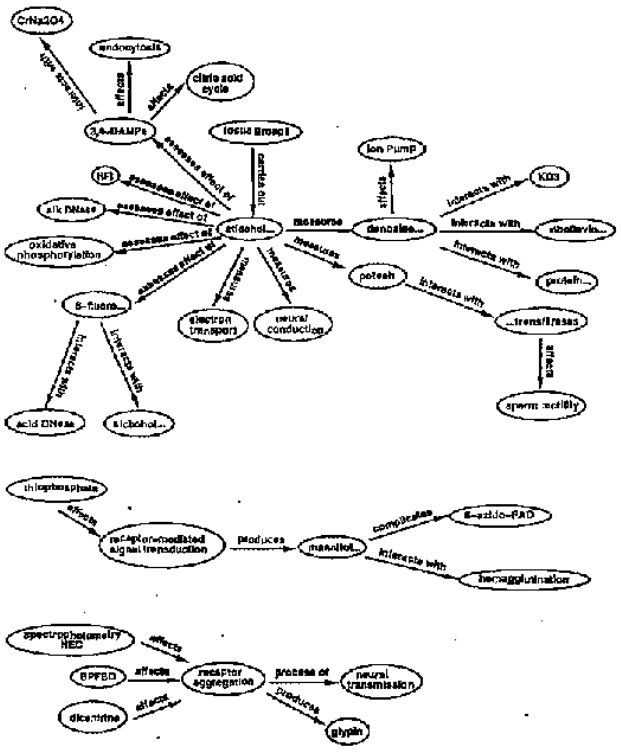
This content label has:

33 vertices
30 edges
464 index terms

Searching for Search Engines



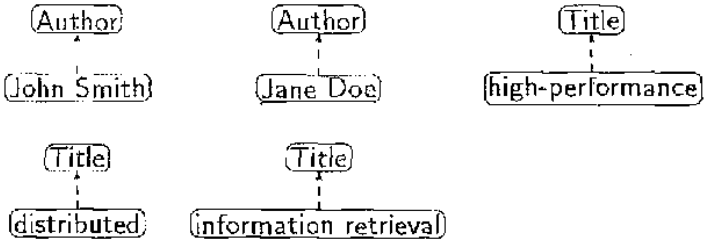
Example of a Medical Content Label



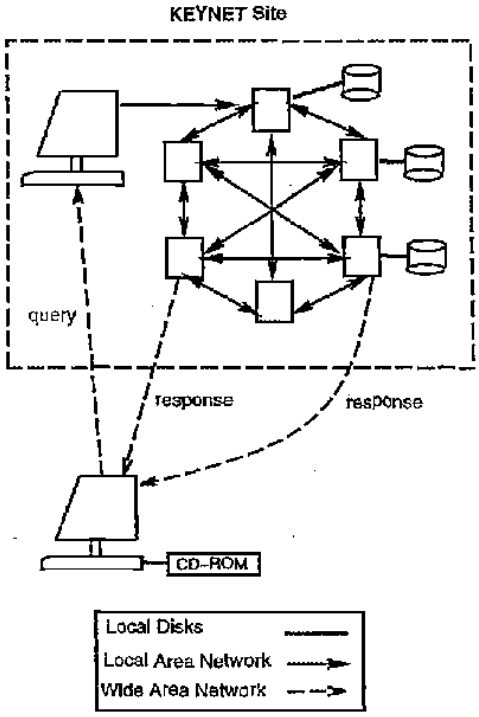
April, 1994

Example of a Content Label

Content label for "High-performance, distributed information retrieval," by John Smith and Jane Doe:



KEYMART: Budget Search Engines



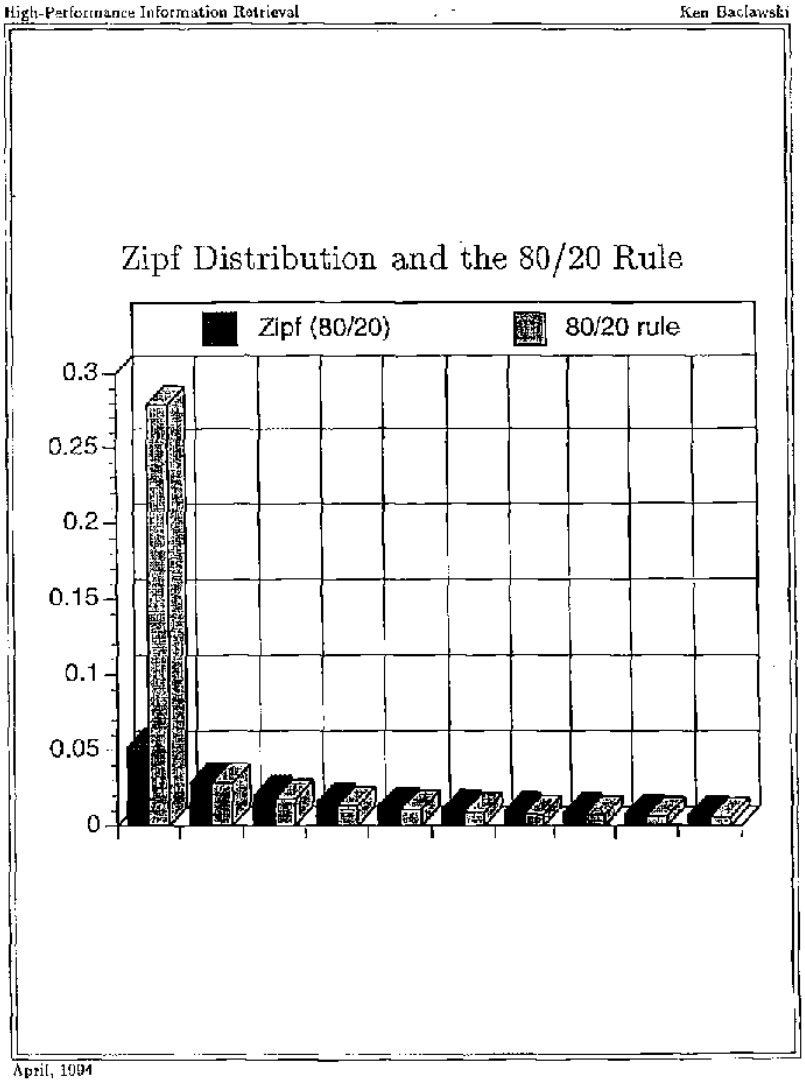
- Unlike the high-performance form of KEYNET, the KEYMART system keeps the index on local disks.

Prototype Specifications

- Processing nodes: Sparcstation 10/30
- RAM: 16MB per processor
- Network: Twisted pair ($\leq$ 100 Mb/sec)
- Content Label: 200 index terms
- Queries: 10 probes (i.e., 10 keywords or attributes)
- Information Objects per processor: 20,000

Prototype Specifications (Continued)

- Memory Model: Local (i.e., "shared nothing")
- Server Model: Each node is a connectionless, multi-threaded, stateless server, and threads can be preempted by higher-priority threads
- Communication Model: One-way datagram service (UDP) using TCP/IP over the Internet
- Indexing Technique: Distributed hashing, not yet expandible
- Best Throughput: 900 queries/second with 1.3 second median response time

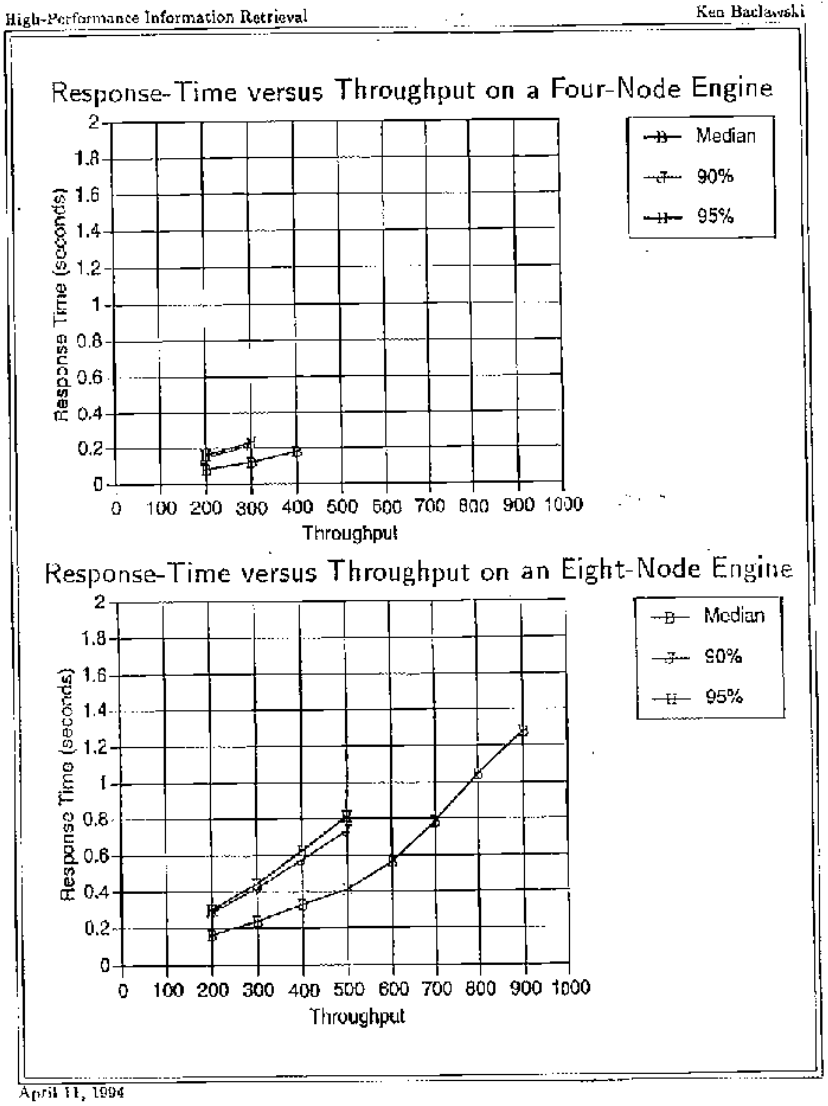

Zipf Distribution and the 80/20 Rule
Zipf (80/20)
80/20 rule
0.3
0.25
0.2
0.15
0.1
0.05
0

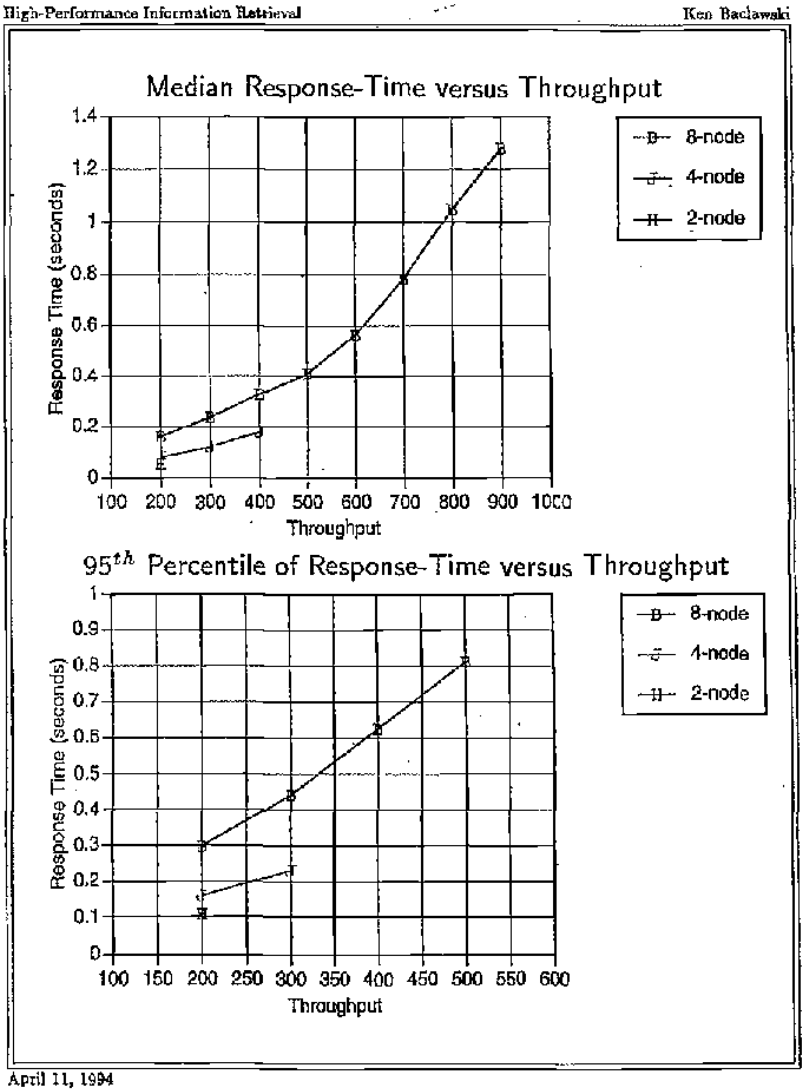
Reliability of KEYNET

KEYNET can tolerate:
- Lost and duplicated messages
- failures of nodes of the network

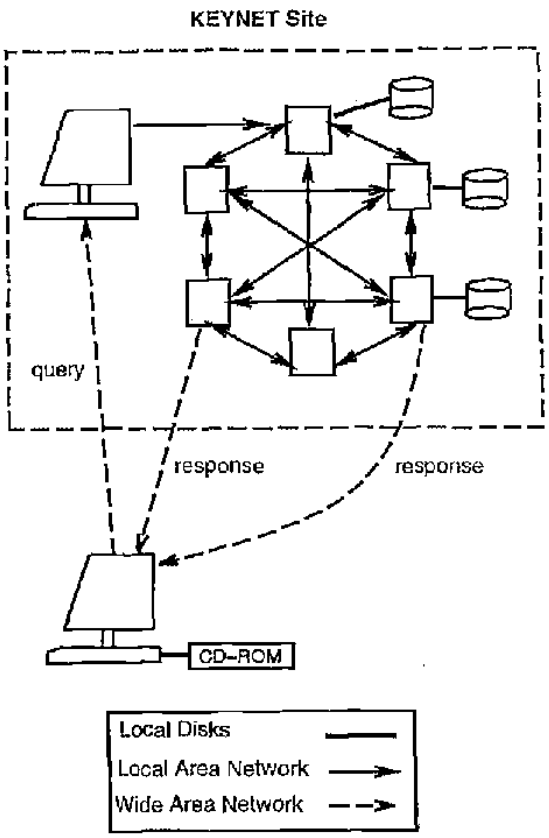
The reason for the reliability of KEYNET is:
- Information retrieval is naturally packet-oriented
  - Queries are generally small
  - Results consist of sets of objects
- Fragmentation assists in reliability
  - Fragments overlap one another. The role played by a lost or duplicated fragment is likely to be compensated by fragments that overlap it
  - Important fragments can be hashed more than once thereby selectively replicating critical information

The reliability of the keynet algorithm allows a connectionless communication protocol can be used. This has much better performance.


Response-Time versus Throughput on a Four-Node Engine
Response Time (seconds)
Throughput
Median
90%
95%
Response-Time versus Throughput on an Eight-Node Engine
Response Time (seconds)
Throughput
Median
90%
95%


Median Response-Time versus Throughput
Response Time (seconds)
Throughput
8-node
4-node
2-node
95th Percentile of Response-Time versus Throughput
Response Time (seconds)
Throughput
8-node
4-node
2-node

Architecture of KEYNET



Keynets unify and extend many information retrieval mechanisms

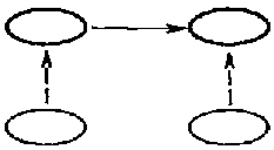
Subject Classification Scheme

I. Biology
- A. Molecular Biology
- B. Cell Biology
- ...

II. Chemistry
- A. Physical Chemistry
- B. Organic Chemistry
- ...

Attributes

Find Author = Einstein
Find Subject = Physics
...



KEYNET

Keywords

Object–oriented
Information retrieval
Distributed ontology
Vector space model

Abstracts / Reviews

A vector space information retreival model is proposed. The proposed model unifies and extends subject classifications, attributes, ...

Ranges and Wildcards

1985 .. 1988
16 .. 18 degrees C
Eins*

Boolean versus Vector Methods

The Boolean Model:

- Queries are boolean expressions
- A Document is a set of terms and attributes
- Documents either satisfy a query or don't satisfy it
- Searching and filtering are completely different

The Vector Model:

- Both queries and documents are vectors
- The components of a vector are terms and attributes
- A document has a degree of relevance to a query
- Searching and filtering are essentially the same

Boolean/Vector Hybrid Models:

- Use vector query as "coarse filter" and boolean query for "fine-tuning"
- Vector queries can be regarded as conjunctions that can be partially satisfied

Related Work

- Information Retrieval

  - Most researchers in IR regard semantic networks as cumbersome, inefficient and suitable only for small databases. The two main models are the vector space model and the boolean model. Most systems and research are designed for textual documents using an unrestricted vocabulary.

    Salton 1989.
    Frakes and Ricardo Baeza-Yates 1992.

  - Some researchers have used knowledge-based indices successfully. None have used semantic networks as the basis for retrieval

    Fuhr *et al.* (AIR/X) in RIAO-91.
    Jacobs in IEEE Expert 1993.

- Semantic Networks

  - Graphical techniques for knowledge representation was first done in 1870 by Peirce. Retrieval methods for semantic networks generally use tree indexes and are limited to very small databases.

    Peirce 1870.
    Levinson in IJIS 1991.
    Lehmann in CMA 1992.
    Levinson in CMA 1992.

- Query Modification

  - The IR community pioneered the use of query modification techniques.

    Chen and Dhar in SIGIR 1990.
    Harman in SIGIR 1992.
    Gauch and Smith in JASIS 1993.
    Qiu and Frei in SIGIR 1993.

  - Database researchers have recently also considered query modification.

    Chu and Chen in JIIS 1992.

- Distributed File Structures

  - Distributed hashing

    Litwin, Neimat and Schneider in SIGMOD Record 1993.

  - Distributed random file generation

    Gray, Sundaresan, Englert, Baclawski and Weinberger in SIGMOD 1994.

- Ontologies

  - There has been a great deal of interest in formal ontologies. The most ambitious effort for building a subject-specific ontology is the UMLS project.

    Humphreys and Lindberg in BMLA 1993.
    Baclawski *et al.* in AAAS 1993.

Automatic Text Processing

The Transformation, Analysis, and Retrieval of Information by Computer

Gerard Salton

Cornell University

ADDISON-WESLEY PUBLISHING COMPANY
Reading, Massachusetts • Menlo Park, California
New York • Don Mills, Ontario • Wokingham, England
Amsterdam • Bonn • Sydney • Singapore
Tokyo • Madrid • San Juan

This book is in the Addison-Wesley Series in Computer Science

Michael A. Harrison, Consulting Editor

Library of Congress Cataloging-in-Publication Data

Salton, Gerard.
Automatic text processing: the transformation, analysis, and retrieval of information by computer / by Gerard Salton.
p. cm.
Bibliography: p.
Includes index.
ISBN 0-201-12227-8
1. Text processing (Computer science) I. Title.
QA76.9.T48S25 1989
005—dc19 88-467
CIP

Reprinted with corrections December, 1988

3 4 5 6 7 8 9 10 HA 95949392

the form

$$D_i = (a_{i1}, a_{i2}, \ldots, a_{it}) \tag{10.1a}$$

and

$$Q_j = (q_{j1}, q_{j2}, \ldots, q_{jt}) \tag{10.1b}$$

where the coefficients a_{ik} and q_{jk} represent the values of term k in document D_i or query Q_j, respectively. Typically a_{ik} (or q_{jk}) is set equal to 1 when term k appears in document D_i (or in query Q_j), and to 0 when the term is absent from the vector. Alternatively, the vector coefficients could take on numeric values, the size of the coefficient depending on the importance of the term in the respective document or query.

Consider now a situation in which t distinct terms are available to characterize record content. Each of the t terms can then be identified with a term vector T_i and a vector space is defined whenever the T vectors are linearly independent. In such a space, any vector can be represented as a linear combination of the t term vectors. Hence the rth document D_r can be written as

$$D_r = \sum_{i=1}^{t} a_{ri}\, T_i \tag{10.2}$$

where the a_{ri}s are interpreted as the components of D_r along the vector T_i. Figure 10.1 represents a typical document vector in a two-dimensional vector space. [3]

In a vector space, the similarity between vectors x and y can be measured by the product $x \cdot y = |x|\,|y| \cos\alpha$, where $|x|$ is the length of x and α is the angle between the two vectors. Hence given a document D_r and a query Q_s represented in the form specified by expression (10.2), the document-query similarity can be computed as

$$D_r \cdot Q_s = \sum_{i=1}^{t} \sum_{j=1}^{t} a_{ri}\, q_{sj}\, T_i \cdot T_j. \tag{10.3}$$

Computing the similarity values of expression (10.3) thus depends on a specification of the document and query components, as well as knowledge of the term correlations $T_i \cdot T_j$ for all term pairs. The vector components are usually generated by an indexing operation from which a term-document matrix is obtained like the one shown in Fig. 10.2 for a collection of N documents. The term correlations, on the other hand, are not usually available a priori, and it is not simple to generate useful term associations.

hms for Automatic
— *Experiments in*
ditor, Prentice-Hall,

Using Relevance —
sification, *Informa-*
–15.

ACM SIGIR Forum,

mation Retrieval, in
in Automatic Docu-
lall Inc., Englewood

ck, in *The Smart Re-*
ocument Processing,
od Cliffs, NJ, 1971,

. Relevance Weights
nentation, 37:4, De

Chapter 10

Advanced Information-Retrieval Models

10.1 The Vector Space Model

❖ 10.1.1 Basic Vector-processing Model

Various mathematical models have been proposed to represent information-retrieval systems and procedures, including the Boolean model, which compares Boolean query statements with the term sets used to identify document content; a probabilistic model based on the computation of relevance probabilities for the documents of a collection; and the *vector-space* model, which represents both queries and documents by term sets and computes global similarities between queries and documents. Of these, the vector-space model is the simplest to use and in some ways the most productive.

The vector-space model assumes that an available term set is used to identify both stored records and information requests. [1,2] Both queries and documents can then be represented as *term vectors* of

(10.4):

$$\text{sim}(D_r, Q_s) = \sum_{i,j=1}^{t} a_{ri}\, q_{si}. \tag{10.4}$$

A similar computation can then be used to obtain pair-wise similarity measurements between documents, the latter forming a basis for certain document-clustering systems:

$$\text{sim}(D_r, D_s) = \sum_{i,j=1}^{t} a_{ri}\, a_{si}. \tag{10.5}$$

When appropriate information is available about the associations that may exist between term pairs — for example, as a result of the pseudoclassification process described in Chapter 9 — the more precise query-document similarity computations of expression (10.3) can, of course, replace the reduced computations of expressions (10.4) and (10.5).

Consider as an example the sample computation of Fig. 10.3. [3] The standard sum-of-products form of Fig. 10.3(c) produces similarity coefficients of 10 and 2 between the query and documents D_1 and D_2, respectively. When the term correlations specified in Fig. 10.3(b) are incorporated, reduced similarity measurements of 8.8 and -0.8 are obtained for the two documents because of the presence in the documents of term T_2, which is negatively correlated with query term T_1.

It should be pointed out that term correlations can in principle be computed from the term-document matrix of Fig. 10.2 by noting that each term is implicitly specified by its assignment to the documents of the collection. In that case term T_i is specified as the ith column of the term-document matrix:

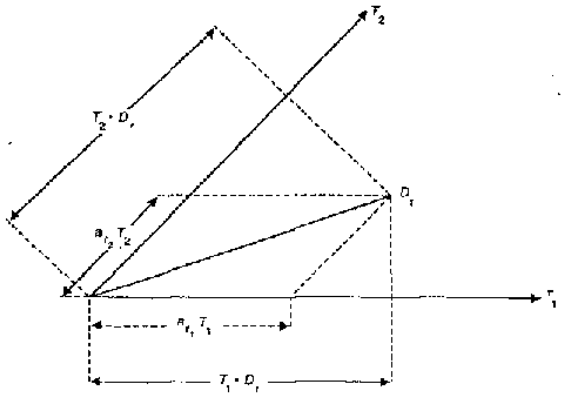
$$T_i = \sum_{r=1}^{N} b_{ir}\, D_r \tag{10.6}$$

The correlation between two columns of the matrix produces a term-term similarity between T_i and T_j as in expression (10.7):

$$T_i \cdot T_j = \sum_{r,s=1}^{N} b_{ir}\, b_{js}\, D_r \cdot D_s. \tag{10.7}$$

To actually obtain the term-term similarities from the data of Fig. 10.2, it is then necessary to assume that the document vectors are

Figure 10.1 Document representation in vector space.



In practice the term-correlation problem is often solved, or circumvented, by assuming that the terms are in fact uncorrelated, in which case the term vectors are orthogonal (that is, $T_i \cdot T_j = 0$, except when $i = j$ and $T_i \cdot T_i = 1$). When the t term vectors are orthogonal, linear independence follows automatically, and the t term vectors form a proper basis for the vector space. Assuming that the terms are uncorrelated, the term-document similarity computation of expression (10.3) is reduced to the simple sum-of-products form of expression

Figure 10.2 Term-document matrix A.

$$A = \begin{array}{c} \\ D_1 \\ D_2 \\ \\ \\ D_N \end{array} \begin{array}{c} \begin{array}{ccc} T_1 & T_2 & T_t \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & a_{1t} \\ a_{21} & a_{22} & a_{2t} \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ a_{N1} & a_{N2} & a_{Nt} \end{bmatrix} \end{array}$$

2. The size of the retrieved set can be adapted to the users' requirements by retrieving only the top few items in the ranked order when casual users are involved, while providing a more exhaustive group of items to specialized users who may require high-recall performance.
3. Items retrieved early in a search, which are most similar to the queries, may help generate improved query formulations using relevance feedback.

A list of typical vector-similarity measures appears in Table 10.1. The first row shows the inner product (or sum-of-products) formula introduced in expressions (10.4) and (10.5). For binary vectors, the inner product measures the number of matching terms in the two vectors, whereas for weighted term vectors, it corresponds to the sum of the products of the weights for equivalent terms. Because this last co-

Table 10.1 Measures of vector similarity

Similarity Measure $sim(X,Y)$	Evaluation for Binary Term Vectors	Evaluation for Weighted Term Vectors
Inner product	$\lvert X \cap Y \rvert$	$\sum_{i=1}^{t} x_i \cdot y_i$
Dice coefficient	$2\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert}$	$\frac{2\sum_{i=1}^{t} x_i y_i}{\sum_{i=1}^{t} x_i^2 + \sum_{i=1}^{t} y_i^2}$
Cosine coefficient	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert^{1/2} \cdot \lvert Y \rvert^{1/2}}$	$\frac{\sum_{i=1}^{t} x_i y_i}{\sqrt{\sum_{i=1}^{t} x_i^2 \cdot \sum_{i=1}^{t} y_i^2}}$
Jaccard coefficient	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert - \lvert X \cap Y \rvert}$	$\frac{\sum_{i=1}^{t} x_i y_i}{\sum_{i=1}^{t} x_i^2 + \sum_{i=1}^{t} y_i^2 - \sum_{i=1}^{t} x_i y_i}$

$X = \{x_1, x_2, \ldots, x_t\}$
$\lvert X \rvert$ = number of terms in X
$\lvert X \cap Y \rvert$ = number of terms appearing jointly in X and Y

themselves orthogonal (that is, $D_r \cdot D_s = 0$ when $r \neq s$). However, such an assumption is clearly unreasonable for any kind of practical document collection. [3]

As a matter of practice, the vector-space model can then be used to obtain correlations, or similarities, between pairs of stored documents, or between queries and documents, under the assumption that the t term vectors are orthogonal, or that the term vectors are linearly independent, so that a proper basis exists for the vector space. When term dependencies or associations are available from outside sources, they can be taken into account (see the sample computations of Fig. 10.3).

There are three main reasons why it is advantageous to generate similarity coefficients between queries and documents in information-retrieval environments:

1. The documents can be arranged in decreasing order of corresponding similarity with the query, making it possible to display retrieved items in decreasing order of presumed importance.

Figure 10.3 Sample query-document similarity computations (from [3]). (a) Sample documents and query. (b) Assumed term correlations (c) Similarity computations for uncorrelated terms. (d) Similarity computations for correlated terms.

(a)

$$D_1 = 2T_1 + 3T_2 + 5T_3$$
$$D_2 = 3T_1 + 7T_2 + 1T_3$$
$$Q = 0T_1 + 0T_2 + 2T_3$$

(b)

	T_1	T_2	T_3
T_1	1	0.5	0
T_2	0.5	1	−0.2
T_3	0	−0.2	1

(c)

$$\text{sim}(D_1, Q) = 2 \cdot 0 + 3 \cdot 0 + 5 \cdot 2 = 10$$
$$\text{sim}(D_2, Q) = 3 \cdot 0 + 7 \cdot 0 + 1 \cdot 2 = 2$$

(d)

$$\begin{aligned} \text{sim}(D_1, Q) &= (2T_1 + 3T_2 + 5T_3) \cdot (2T_3) \\ &= 4T_1 \cdot T_3 + 6T_2 \cdot T_3 + 10T_3 \cdot T_3 \\ &= -6 \cdot 0.2 + 10 \cdot 1 \\ &= 8.8 \end{aligned}$$

$$\begin{aligned} \text{sim}(D_2, Q) &= (3T_1 + 7T_2 + 1T_3) \cdot (2T_3) \\ &= 6T_1 \cdot T_3 + 14T_2 \cdot T_3 + 2T_3 \cdot T_3 \\ &= -14 \cdot 0.2 + 2 \cdot 1 \\ &= -0.8 \end{aligned}$$

tions such a query has the form

$$Q_{opt} = k\left\{\frac{1}{R}\sum_{Rel}\frac{D_i}{|D_i|} - \frac{1}{N-R}\sum_{Nonrel}\frac{D_i}{|D_i|}\right\}, \tag{10.8}$$

where R and $N-R$ are the assumed number of relevant and nonrelevant documents, and the two summations range over the sets of normalized relevant and nonrelevant documents, respectively. [5]

The summation of expression (10.8) is not immediately useful for constructing queries because the sets of relevant and nonrelevant documents with respect to the queries are not known, of course, before an exhaustive search. Hence the relevant and nonrelevant document vectors cannot be summed to construct the desired query vector. Expression (10.8) can be approximated after an initial search operation, however, by asking the user to assess the relevance of some of the previously retrieved items. This identifies some subset R' of the R items relevant to the query, and some subset N' of the $N-R$ nonrelevant documents. An approximation of the optimal formulation of expression (10.8) is then obtained by taking an available query formulation Q^0 and adding the vector elements for the items identified as relevant, while subtracting the vectors for the items identified as nonrelevant:

$$Q^{(j+1)} = Q^{(j)} + \frac{1}{|R'|}\sum_{D_i \in R'} D_i - \frac{1}{|N'|}\sum_{D_i \in N'} D_i \tag{10.9}$$

or alternatively,

$$Q^{(j+1)} = Q^{(j)} + \alpha \sum_{D_i \in R'} D_i - \beta \sum_{D_i \in N'} D_i \tag{10.10}$$

for suitable multipliers α and β.

As indicated by the form of expressions (10.9) and (10.10), the steps in relevance feedback can be carried out iteratively in such a way that the query statement is made to approach the optimal query little by little as the relevance status of more and more documents becomes known. Also, multipliers α and β in (10.10) can be chosen in various ways to reflect techniques that give equal weight to relevant and nonrelevant items ($\alpha = \beta = 0.5$), or that favor only the relevant items ($\alpha = 1$, $\beta = 0$). An especially effective feedback technique is the so-called "dec hi" method, which uses all positive (that is, relevant) information for feedback purposes but subtracts only the highest ranked nonrelevant document. [6,7]

Figure 10.4(a) illustrates a positive relevance-feedback technique, with the t-dimensional vector space rendered in two dimensions, and

efficient is in principle unbounded, it is customary in most applications to use normalized similarity coefficients whose values vary between 0 and 1 when the vector elements are nonnegative. Three typical normalized similarity coefficients of this kind are the Dice, cosine, and Jaccard coefficients, also included in the table. [4]

The disadvantages of the basic vector-processing model relate to the assumed orthogonality, and hence independence between terms, and the lack of theoretical justification for some of the vector-manipulation operations. For example, the choice of a particular vector-similarity measure for a certain application is not prescribed by any theoretical considerations, and is left to the user. Some of the advantages are the model's simplicity, the ease with which it accommodates weighted terms, and its provision of ranked retrieval output in decreasing order of query-document similarity. A main virtue of this approach is the ease with which individual vectors can be modified, making it possible to adapt the query and document vectors to a dynamic environment as described in the next subsection.

10.1.2 Vector Modifications

One of the most important and difficult operations in information retrieval is generating useful query statements that can extract materials wanted by users and reject the remainder. Since usually an ideal query representation cannot be generated without knowing a great deal about the composition of the collection, it is customary to conduct searches iteratively, first operating with a tentative query formulation, and then improving formulations for subsequent searches based on evaluations of the previously retrieved materials. One method for automatically generating improved query formulations is the well-known relevance-feedback process. [5–7]

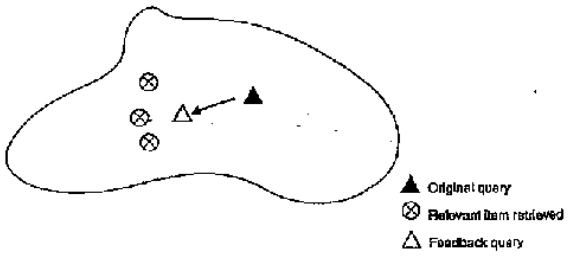
The main assumption behind relevance feedback is that documents relevant to a particular query resemble each other in the sense that they are represented by reasonably similar vectors. This implies that if a retrieved document has been identified as relevant to a given query, the query formulation can be improved by increasing its similarity to such a previously retrieved relevant item. The reformulated query is expected to retrieve additional relevant items that are similar to the originally identified relevant item.

It remains to find an effective method for "moving" a given query toward the relevant items and away from the nonrelevant ones. Consider first the construction of an ideal query which maximizes the average query document similarity for the relevant documents, at the same time minimizing the average query-document similarity for the nonrelevant documents. It is known that under appropriate assump-

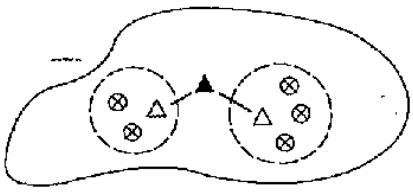
the distance between two items assumed to be inversely proportional to the similarity between corresponding vectors. The transformation of expression (10.10) effectively moves the query toward the area of the vector space where the relevant items are located.

In some cases, the basic relevance feedback process does not operate satisfactorily because the relevant documents identified do not form a tight cluster in the document space, or because nonrelevant documents are scattered among certain relevant ones. One way to detect such conditions is by clustering the previously identified relevant items and calling for remedial action whenever more than one homo-

Figure 10.4 Relevance-feedback illustration. (a) Positive relevance feedback. (b) Feedback with query splitting.



(a)



(b)

geneous relevant document cluster is detected. Such action could consist of splitting the original query into several pieces by constructing *one new* feedback query for each subset of identified, homogeneous relevant items. Figure 10.4(b) illustrates such a *query-splitting* operation; two new feedback queries are generated from a single original query because the previously identified relevant items are *separated* into two subgroups. [8]

By assuming that the query-document and the document-document similarity coefficients for a collection follow certain specified probabilistic assumptions, formal conditions can be given for the construction of improved feedback queries using relevance feedback. [9] A single iteration of relevance feedback usually produces improvements of from 40 to 60 percent in the search precision, evaluated at certain fixed levels of the recall and averaged over a number of user queries. [10,11] In evaluating the relevance feedback process, care must be taken to disregard the effect of gains in recall and precision due to improved ranking in the retrieval of relevant documents retrieved in earlier search iterations, and subsequently used for feedback. Thus in Fig. 10.5(a) two documents originally retrieved in ranks 2 and 5 were used to construct the feedback query. Since the feedback process forces the new query to approach to two previously retrieved documents, these items will probably be retrieved in the top two ranks in a new search conducted with the reformulated query. The rank improvement from 2 and 5 to 1 and 2 is not of genuine interest, however, because the user has already seen these two relevant items after the original search, and hence does not profit from the subsequent retrieval with improved ranks.

Various methods exist for disregarding this ranking effect. One method, residual collection evaluation with partial rank freezing, is illustrated in Fig. 10.5. In this method, the previously retrieved items identified as relevant are kept "frozen" in their original retrieval ranks, and the previously retrieved nonrelevant items are simply removed from the collection. their ranks (1, 3, and 4 in Fig. 10.5a) occupied by items that are newly retrieved in subsequent search iterations. [12] Recall and precision values are computed after each document is retrieved; the corresponding values are plotted in Fig. 10.5(c). The ranks of the relevant items in the figure improve from 2, 5, 7, and 8 to 2, 3, 5, 6, and 7.

Another method for evaluating relevance feedback consists of splitting the document collection into two pieces, taking care to even out the relevance properties of the two halves (the average number of relevant items per query). The test-collection half is then used to construct modified feedback queries using the normal relevance-feedback process, while the control half serves for the evaluation (see Fig. 10.6). [12]

Figure 10.5 Residual collection evaluation with partial rank freezing. (a) Initial query performance. (b) Feedback query performance. (c) Recall-precision graph.

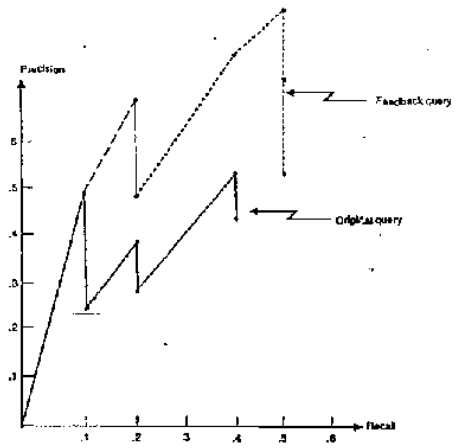
Rank	Document number	Recall	Precision
1	x 1	0	0
2	• 2	1/10	1/2
3	x 3	1/10	1/3
4	x 4	1/10	1/4
5	• 5	2/10	2/5
Feedback cutoff			
6	x 6	2/10	2/6
7	• 7	3/10	3/7
8	• 8	4/10	4/8
9	x 9	4/10	4/9
10	x 10	4/10	4/10

(a)

Rank	Document number	Recall	Precision
1	x 4	0	0
2	• 1	1/10	1/2
3	• 9	2/10	2/3
4	x 5	2/10	2/4
5	• 2	3/10	3/5
6	• 8	4/10	4/6
7	• 5	5/10	5/7
8	x 6	5/10	5/8
9	x 7	5/10	5/9
10	x 8	5/10	5/10

• Relevant items
x Nonrelevant items

(b)

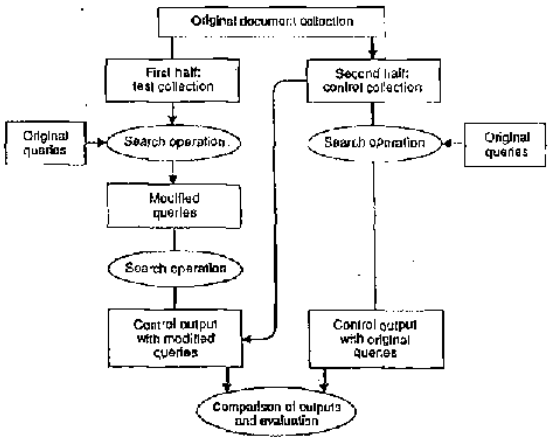


(c)

The test-and-control evaluation method poses fewer conceptual problems than residual-collection evaluation, but splitting the collection into halves with similar relevance properties is not always easy to do.

The relevance feedback process generates improved query formulations without modifying the document vectors themselves. A similar approach, however, can be used to also alter the document vectors in the course of operations. Such a *document-space modification* operation improves document indexes, reflecting the experiences of prior collection processing. [13,14] In one such operation, the vectors of documents previously retrieved in response to a common query are modified by moving documents identified as relevant closer to the common query, and at the same time moving documents identified as nonrelevant away from the query. This operation is represented in simplified form in Fig. 10.7. It is clear from the illustration that when the relevant items approach a common query location, these documents will resemble each other more closely than before, and therefore can be retrieved more easily in response to a similar query submitted lat-

Figure 10.6 Test-and-control collection evaluation.



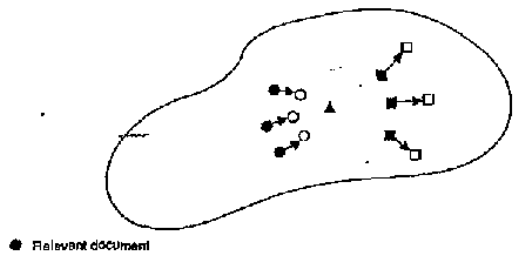
er. The converse is true for the nonrelevant documents that are shifted away from the query.

The easiest way to modify the vectors of the desired documents is to add the terms from the query vector to the documents previously identified as relevant, and subtract the query terms from the documents identified as nonrelevant. [13] Care must be taken to control the movements of the documents in the vector space. Since relevance assessments by users are necessarily subjective, each individual document movement must remain small, ensuring that substantial changes in document position reflect the effects of many small individual modifications performed by different users. In practice, then, only small modifications of the term weights are allowed at each iteration.

The automatic document-space-modification process can be used to control a collection by "promoting" documents deemed important, at the same time "demoting" those considered extraneous. Such an approach serves to control a collection's growth by identifying unimportant items — those never requested during previous searches — and eventually removing them from the collection, or shifting them to an auxiliary portion of the files. [15,16]

Various document-retirement algorithms have been used experimentally, including procedures in which the weights of all promotable documents — those retrieved early in a search, or deemed relevant by users — are multiplied by a small positive quantity in each search iteration, while demotable documents — those retrieved at the bottom of

Figure 10.7 Document-space modification.



the ranked output lists — are multiplied by a small negative quantity. Instead of altering the term weights directly, a separate usage indicator attached to each document can be modified appropriately. In any case, documents are retired from the collection when the value of the document usage indicator, or the average term weight, falls below a given threshold. Retirement algorithms can also be based on conventional document-space-modification methods of the kind illustrated in Fig. 10.7, in which items that have been moved out to the far periphery of the document space are eliminated.

Document-space-modification methods are difficult to evaluate in the laboratory, where no users are available to dynamically control the space modifications by submitting queries. However, indications are that substantial growth in a collection can be accommodated without requiring basic changes in the collection's organization, and that document-retirement rates of up to 10 percent will affect search recall and precision only minimally. [15,16]

10.2 Automatic Document Classification

10.2.1 General Considerations

A disadvantage of conventional information-retrieval methodology using inverted index files, is that the information pertaining to a document is scattered among many different inverted-term lists. Further information relating to different documents with similar term assignments is not in close proximity in the file system. If browsing is to be permitted in the document collection, however, data about related items should appear close together. Browsing operations become possible when collection grouping, or *clustering* operations, group into common areas items judged to be similar. Clustered document collections can be maintained in addition to the normal inverted-index files. In that case, collection searches can be conducted conventionally, and browsing operations carried out using the special cluster structure. Alternatively, inverted indexes may be completely abandoned when a clustered file organization is available — the cluster structure can in fact be used for both searching and browsing.

In general, a clustered file provides efficient file access by limiting the searches to those document clusters which appear to be most similar to the corresponding queries. At the same time, clustered file searches may also be effective in retrieving the wanted items whenever the so-called *cluster hypothesis* holds — that is, when associations between documents, measured by document-document similarity coefficients, convey information about the joint relevance of documents to queries in a collection. [4,17,18]

The outline of a typical clustered file appears in Fig. 10.8, in which the main file is assumed to be grouped into three large clusters, each subdivided into a number of smaller, lower-level clusters that in turn contain individual documents, each represented by an *x*. Depending on the subject areas covered, a collection may be divided into many different clustering levels, instead of only the three levels in Fig. 10.8. In a clustered file, each cluster, or supercluster, can be represented by a special term vector, known as the *cluster centroid*. The centroid may actually be identical to a particular document included in the corresponding cluster; more usually it is a special dummy item separately constructed from the remaining vectors in the cluster and located in the center of the corresponding cluster. The centroids in Fig. 10.8 include a hypercentroid, representing the center for the complete space; supercentroids, representing the next level of the structure; and the lower-level centroids, representing regular clusters. Using a file organization like the one shown in the figure, a file search can be conducted by comparing the query first against the highest-level centroids (supercentroids in the figure). For higher-level centroids that prove sufficiently similar to the query, a query-centroid comparison is then carried out with the lower-level centroids. This operation is re-

Figure 10.8 Typical clustered file organization.

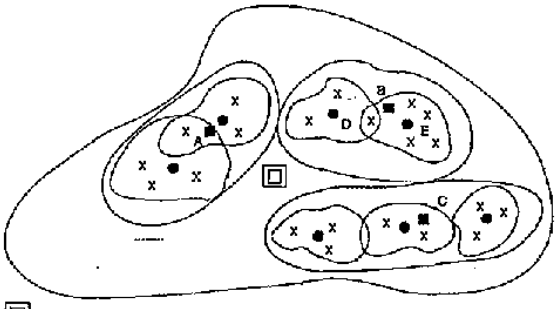
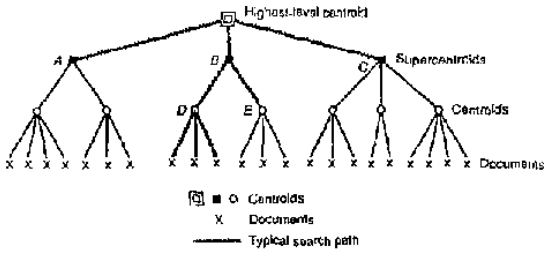


Figure 10.9 Search strategy for clustered file of Figure 10.8.



peated until eventually some actual documents are identified that are included in some of the low-level clusters.

A search tree for the cluster organization of Fig. 10.8 is shown in Fig. 10.9. A typical search carried out with this cluster structure first compares the query with the three supercentroids, labeled A, B, and C in Fig. 10.9. Assuming that supercentroid B is similar to the sample query, the next comparison is made with centroids D and E, located in the supercluster represented by supercentroid B. If D now matches the query closely enough, the three documents in the corresponding cluster are processed, and any document whose query-document similarity exceeds the established similarity threshold is presented to the user.

In discussing the use of clustered files, it is convenient to distinguish the *cluster-generation* process from the *cluster-search* strategies. In normal circumstances, the cluster structure is generated only once, and cluster maintenance can be carried out at relatively infrequent intervals. Cluster-search operations, on the other hand, may have to be performed continually. These operations must therefore be carried out efficiently, even while the generation process may be slower and relatively more expensive. [19]

10.2.2 Hierarchical Cluster Generation

Two main strategies can be used to carry out the cluster generation. First, a complete list of all pairwise item similarities can be constructed, in which case it is necessary to employ a grouping mechanism capable of assembling into common clusters items with sufficiently large pairwise similarities. Alternatively, *heuristic methods* can be

AUTOMATIC STRUCTURING AND RETRIEVAL OF LARGE TEXT FILES

Gerard Salton, James Allan, and Chris Buckley

In many practical situations it is necessary to deal with large heterogeneous collections of text. Most notably, this is the case for newspaper files, message collections, dictionaries and encyclopedias, textbook materials, and in many library environments. In such situations the subject matter varies widely—often covering large, open-ended slices of knowledge. Normally, selective access is desired to particular items on demand, and file access may be considerably simplified when browsing capabilities allowing a flexible traversal of the text structure are made available.

The conventional wisdom is that sophisticated conceptual text representations are needed for information retrieval purposes, including the use of thesauruses tailored to particular subject areas, and of preconstructed knowledge structures that classify the main entities of interest in a given subject area, and specify the relationships that may hold between the entities in particular areas of application. However, many conceptual and practical problems arise when deep language analysis systems are considered in unrestricted environments with arbitrary subject matter. Viable methods still do not exist for building large thesauruses automatically. And the use of large knowledge bases is hampered by the lack of clarity about what knowledge is actually needed for particular applications, how to represent the needed knowledge, how to isolate individual pieces of knowledge from an apparently unlimited context, and how effectively to match the content of any existing knowledge base with the available text collections.

Fortunately, alternative approaches exist for dealing with large classes of heterogeneous text. When the text collections are available in machine-readable form, corpus-based text-handling systems may be implemented that use the text collections themselves to derive the information necessary for analysis and text characterization. Specifically, the available texts can be processed, and the text words and expressions can be examined in detail, including the contexts in which the text elements are used, the decomposition of the text into sections, paragraphs and sentences, the structure of these various text components, and so on. By judiciously combining information obtained from the available texts, reasonable text characterizations can then be generated.

In this study, procedures are described for analyzing text content, structuring the text by linking text excerpts covering related subject matter, and retrieving text items in response to available user queries. For experimental purposes, the 25,000 articles included in the 29 volumes of the *Funk and Wagnalls New Encyclopedia* (65MB of text) are used as a database [7].

Standard Vector Processing Model and Automatic Text Analysis

In conventional information retrieval environments, keywords are manually or automatically assigned to the information items, and queries are formulated by using terms interconnected by Boolean operators. Although widely used, the Boolean retrieval model is not ideally suited to the retrieval task: users find it difficult to generate effective Boolean queries that will retrieve just the right type and amount of information; the retrieved items are presented to the users in a random order that does not correspond to any presumed order of relevance or usefulness; and term weights reflecting term importance are awkward to incorporate into Boolean systems in a

consistent way. Most important, the operations of Boolean logic are unforgiving and inflexible, and the retrieval results are often inadequate [13, 14, 22].

The vector processing model represents an alternative possibility for handling information retrieval operations. In that case, both the stored documents as well as the search requests are represented by sets of terms (term vectors) without Boolean operators. Different vectors can be compared with each other and vector similarity coefficients are obtainable reflecting similarity in the term assignments for different vectors. In the vector processing model of retrieval, the same methods are usable for collection structuring (by comparing pairs of document vectors with each other and identifying document pairs found to be sufficiently similar), and for information retrieval (by comparing query vectors with the vectors representing the stored items and retrieving items found to be similar to the queries).

The results of a similarity computation between a query vector and the stored document vectors can be ranked in decreasing order of the computed query similarity. This makes it possible to retrieve the most important items (those most similar to the user queries) first. Furthermore, term weights are easily accommodated because vectors of weighted terms are manipulated almost as easily as binary term vectors where weights are restricted to 1 for assigned terms and 0 for missing terms [12, 29].

A high-performance term weighting system assigns large weights to terms that occur frequently in particular documents, but rarely on the outside, because such terms are able to distinguish the items in which they occur from the remainder of the collection.

A typical term weight of this type, known as a *tf* × *idf* weight (term frequency times inverse document frequency) may be defined as

$$w_{ik} = \frac{tf_{ik} \cdot \log(N/n_k)}{\sqrt{\sum_{j=1}^{t} (tf_{ij})^2 \cdot (\log(N/n_j))^2}} \qquad (1)$$

where w_{ik} represents the weight of term T_k assigned to document D_i, tf_{ik} is the frequency of occurrence of term T_k in D_i (a frequency of 0 is assumed for terms not assigned to D_i), N is the size of the document collection, and n_k represents the number of documents in the collection with T_k. The summation in the denominator of expression (1), taken over all terms in a particular vector, is used for length normalization to ensure that all documents have equal chance of being retrieved. (Without length normalization, the longer documents with more assigned terms and higher term frequencies would generate higher document similarities, and exhibit higher retrieval potential, than the shorter items.) [18].

Given query and document vectors $Q_j = (w_{j1}, w_{j2}, \ldots, w_{jt})$ and $D_i = (w_{i1}, w_{i2}, \ldots, w_{it})$, respectively, or alternatively, given two document vectors D_j and D_i a vector similarity function of the form

$$sim(Q_j, D_i) = \sum_{k=1}^{t} w_{jk} w_{ik} \qquad (2)$$

may be used reflecting the similarity between the corresponding term vectors. When normalized term weights are used, such as those of expression (1), the vector similarity lies between 0 and 1 and depends on the proportion and the weight of matching terms in the vectors.

It is well known that most text words and expressions are highly ambiguous when considered out of context. On the other hand, principles are also at work that limit the potential for ambiguous interpretation, at least in ordinary discourse and nonliterary writing. The need

Table 1. Effect of search restrictions

Document number	Query similarity	Title of retrieved item	Rejected by local search restrictions
6008	1.00	Delian League	-
1667	0.58	Athenian League	√ cross-reference
7011	0.49	Delos, Confederation of	-
17572	0.45	Pallas Athena	√ cross-reference
1669	0.43	Athens (Greece)	-
13779	0.43	League of Nations	√ extraneous topic
7010	0.41	Delos	-
10581	0.40	Greece	-
17580	0.39	Pericles	-
1104	0.36	Antalkidas	-
1390	0.35	Aristides	-
14055	0.34	Marathon	√ marginal topic
1663	0.34	Athena	-
5479	0.34	Cimon	-
580	0.33	Alcibiades	-
23226	0.33	Urban League	√ extraneous topic
1274	0.32	Arab League	√ extraneous topic
911	0.32	Amphictyonic League	-

a. Unrestricted search output for article (6008) Delian League

Document number	Query similarity	Title of retrieved item	Some retrieved items in restricted search
6998	1.00	Delian League	√
20011	0.49	Delos, Confederation of	√
1669	0.43	Athens (Greece)	√
20010	0.41	Delos	√
10381	0.40	Greece	√
17980	0.39	Pericles	√
1104	0.36	Antalcidas	√
1330	0.35	Aristides	√
1665	0.34	Athens	√
5479	0.34	Cimon	√
580	0.33	Alcibiades	√
911	0.32	Amphictyonic League	√
20328	0.31	Samos	√
22364	0.30	Themistocles	√
22466	0.30	Thucydides	√
14557	0.30	Lysander	√
3491	0.29	Brasidas	√
16369	0.28	Pisistratus	√

5. Restricted search output for article [6998] Delian League (one sentence match required at threshold 75.0) (Items affected by local match system are marked by √; unaffected items are labeled by -)

for writers and readers to communicate regularizes the language to some extent, and favors simpler languages and interpretations over more complex ones [9]. Consequently, much of the language ambiguity disappears in ordinary discourse when the wider linguistic, social, and temporal contexts are taken into account.

The environment in which text units and complete texts are embedded also plays a major role in the influential *use theory* of meaning proposed by Wittgenstein and others. The following quotation is reflective of this point of view [25].

"For a large class of cases—though not for all—in which we employ the word 'meaning' it can be defined thus: the meaning of a word is its use in the language."

The use theory seems especially appropriate in information retrieval in which the major concern is not with the intrinsic meaning of the words and text units in isolation but with the global meaning of complete text entities. In the retrieval environment it is not normally necessary to assign specific semantic interpretations to individual text words. Instead, it suffices to determine whether different texts—for example, a query text and the texts of stored documents—are close enough to be relatable, that is, whether text use and text environments are congruent.

In practice, it is not always easy to determine the precise purpose and social environment in which a given text is placed. It is, however, normally possible to study the linguistic context in which the words occur. If one assumes that word meanings are related when text words and expressions appear in similar local contexts, the following refined method of text comparison suggests itself: [20, 21].

1. A global text similarity is first computed by comparing the respective text vectors as previously explained. Text pairs without sufficient global similarity are not considered.
2. The local text environments are considered next for texts with sufficient global similarity, and local structures, such as text sections, paragraphs and sentences are compared. When globally similar text pairs also contain a sufficient number of locally similar substructures, the texts are assumed to be related.

The global-local text comparison system should produce a high degree of retrieval precision, that is, very few semantically dissimilar texts will pass the multiple vocabulary filter. A somewhat informal retrieval test based on 800 documents retrieved in response to 100 query texts, produced a retrieval precision of 94.8%, indicating that only about 5% of the retrieved items were extraneous [16]. The recall which measures the proportion of relevant materials actually obtained from the collection cannot be measured precisely in practical *retrieval environments* because the complete set of relevant documents included in a collection is not normally known. In principle, different texts may exist that cover substantially related subject matter in completely different terms. In that case, a vocabulary comparison method is not adequate for text identification. In practice, identical events are not easily described without using substantially overlapping vocabularies. Such an overlap is then detectable by properly designed text-matching methods.

Text Retrieval Operations

The information retrieval operations are illustrated by searches in the *Funk and Wagnalls New Encyclopedia*. In each case, an existing encyclopedia article is used as a search request, and other related articles are then retrieved. Alternatively, a discursive,

Table 2. Relevance feedback illustration (restricted searches requiring one matching sentence pair at threshold 75.0)

Document number	Query similarity	Title of retrieved document
6998	1.00	Delian League
7011	0.49	Delos, Confederation of
1669 Y	0.43	Athens (Greece)
7010	0.41	Delos
10981	0.40	Greece
12980	0.39	Pericles
1104	0.35	Aristides
1390	0.35	Aristides

a. Initially retrieved documents in response to [6998] "Delian League". Y: user indicates that item [1669] "Athens" is important.

Document number	Query similarity	Title of retrieved document
17547	0.48	Parthenon
18359	0.48	Pisistratus
4175	0.45	Callistratus
3491	0.38	Brasidas
10382	0.37	Greek Art and Architecture
22254	0.36	Salamis
22350	0.36	Thebes (Greece)
180	0.36	Acropolis
18431	0.34	Plataea

b. Feedback search incorporating terms from [1669] "Athens" in the reformulated query

natural language query formulation can be used when an appropriate encyclopedia text is not available. A typical encyclopedia search is illustrated in Table 1. The query used is article [6998] Delian League. (The Delian League was a league of Greek city-states founded in the fifth century B.C. to guard against attack from the Persians.)

Table 1a shows the results of an unrestricted search, using the global vector comparisons only (step 1 of the process outlined earlier). The retrieved items are listed in decreasing order of the global query similarity, giving for each retrieved item the document number, the similarity coefficient with the query, the title of the retrieved item, and finally a comment for some of the items. As expected, the top retrieved item in response to query [6998] is the article "Delian League" itself with a perfect similarity of 1.00.

Many of the articles obtained by using the global vector comparison of Table 1a represent Greek topics related to the Delian League. However, a few extraneous articles appear on the list, including notably [12779] League of Nations, [23226] Urban League, and [1274] Arab League. All of these are relatively modern creations of the twentieth century, not related to the Delian League. In addition, a couple of short cross-reference articles of the form "A, see B" are also retrieved by the unrestricted search of Table 1a, including articles [1667] and [17572]. *Such cross-reference articles may* be of marginal interest because they do not contain primary information, but refer instead to other encyclopedia entries.

A restricted search for query [6998] is illustrated in Table 1b. This search involves both the global vector match, as well as a local context check (steps 1 and 2 of the process outlined earlier). In this case, an article must exhibit at least one matching text sentence with the query article before the item is actually retrieved with a pairwise sentence similarity of at least 75.0.

The sentence similarities are obtained by generating a term vector for each sentence included in the given document pair, and computing the vector pair similarity of equation (2). For the sentence pair comparisons, unnormalized term weights are used: that is, the $tf \times idf$ term weighting formula of expression (1) is used without the normalization factor in the denominator. When unnormalized term weights are used, the text similarity depends on the number (rather than the proportion) of matching terms. This implies the similarities between very short sentences which often prove to be meaningless ("consider the following Figure," "see the example in figure x," "consider Figure x," and so forth) will not be treated as significant. Because of the unnormalized term weights, the computed sentence similarity coefficients are normally much larger than 1 [16, 20, 21].

The output of Table 1b shows the questionable articles dealing with the extraneous "leagues" have now disappeared. The same is true of the cross-reference articles listed on the original output. None of these items had any matching sentences with the text of query [6998]. The missing output from Table 1a is replaced by other articles dealing with Greek topics at the bottom of Table 1b.

If the user is not satisfied with the

results of an initial search, an improved query can be obtained using the well-known relevance feedback process [11, 19]. In relevance feedback mode the user designates some *previously retrieved articles as especially useful*. A new query is then automatically generated as a combination of the original query text plus the texts of the relevant items identified by the user. A relevance feedback run is illustrated in Table 2, where the originally retrieved output of Table 2a corresponds to the top of the list for the restricted search of Table 1b. Item [1669] Athens is identified as important (Y = yes), and a new combined query now produces the output of Table 2b, consisting of Athenian topics replacing the originally retrieved [6998] Delian League and [7010] Delos. The relevance feedback process introduced in the 1980s is known to be effective, and is now used in many operational retrieval environments [19, 24].

The encyclopedia articles are subdivided into sections and subsections, each identified by an appropriate section heading, and sections are subdivided in turn into paragraphs and sentences. A topic description can therefore be refined by using article sections, subsections, or paragraphs to formulate the queries, and retrieving article excerpts instead of full articles. A typical hierarchical decomposition, using increasingly more refined query formulations is shown in Table 3. A basic search with the full article [10381] Greece retrieves other articles dealing with Greek topics, such as [1669] Athens, [11109] Hellas, and [11113] Hellenism, as shown in Table 3a. Increasingly more refined searches are illustrated in Tables 3b, 3c, and 3d, using queries Greece/History, Greece/History/Ancient Greece, and Greece/History/Ancient Greece/The ascendancy of Athens, respectively. The slash (/) designates a subdivision into sections or subsections, respectively. Each of the section queries is used to retrieve other text sections in the output of Table 3. As expected, the search conducted with section [10381.c99] (section 99 of document [10381] on Greek History) recovers other sections dealing with aspects of

Table 3. Examples of hierarchical expansion using narrower contexts

Section number	Document number	Query similarity	Document title
10381		1.00	Greece
1669		0.47	Athens (Greece)
11109		0.43	Hellas
15511		0.42	Metaxas, Ioannes
11113		0.41	Hellenism
6998		0.40	Delian League

a. *Search results for full article [10381] "Greece" (at least one sentence required at threshold 75.0)*

Section number	Document number	Query similarity	Document title
76618	10381.c80	1.00	Greece
76622	10381.c48	0.32	Greece/ . . . /Hellenic Period/ Shifting Alliances
76579	10381.c3	0.29	Greece
76628	10381.c54	0.27	Greece/History/ . . . /Greek Renaissance
76620	10381.c46	0.26	Greece/History/ . . . /The ascendancy of Athens
76621	10381.c47	0.26	Greece/History/ . . . /The Peloponnesian War

b. *Search results for section [76615] "Greece/History" (at least one matching sentence required at threshold 75.0)*

Section number	Document number	Query similarity	Document title
76615	10381.c41	1.00	Greece/History/Ancient Greece
54487	9416.c3	0.48	Dorians
76614	10381.c40	0.39	Greece/History/Prehistoric Period
1084	149.c3	0.36	Achaeans
1026	1481.c3	0.35	Achaean League
99290	12235.c3	0.32	Ionians

c. *Search results for section [76615] "Greece/History/Ancient Greece" (at least one matching sentence required at threshold 75.0)*

Section number	Document number	Query similarity	Document title
76620	10381.c46	1.00	Greece/History/ . . . / The Ascendancy of Athens
193112	17980.c3	0.62	Pericles
51079	6998.c3	0.59	Delian League
12476	1669.c7	0.57	Athens/History/The Classical Period
76621	10381.c47	0.51	Greece/History/ . . . / The Peloponnesian War
76618	10381.c44	0.51	Greece/History/ . . . / From Monarchy to Democracy

d. *Search results for section [76620] "Greece/History/Ancient Greece/The Ascendancy of Athens" (at least one matching sentence at 75.0)*

Greek history. When a more refined query dealing with the history of Ancient Greece [1058], section 41] is processed, the retrieved items include articles describing the peoples of Ancient Greece, including the Dorians [9446.e8], the Achaeans [149.e8], and the Ionians [12283.e8]. Finally, the subsection covering the ascendancy of Athens in the history of ancient Greece retrieves other text sections dealing with the role of Athens in ancient Greek history.

In the illustration of Table 3, text sections are retrieved in answer to section queries. For current purposes, a text section is defined as a text segment located between two adjacent section heads. In many cases, a section consists of a single text paragraph treating a unified subject. Alternatively the larger sections may be subdivided into many paragraphs and a variety of different topics may be treated. This suggests that a mixed retrieval strategy be used which retrieves variously full text items, text sections, or text paragraphs, *depending on which of these* text entities exhibits the highest query similarity. For short articles, the full text would normally be obtained. Longer, more discursive texts covering a variety of subjects would be subdivided and only certain relevant sections or paragraphs would be retrieved.

A typical output product of such a mixed retrieval strategy is shown in Table 4, representing the items retrieved in answer to an article about the fourteenth century Italian painter Giotto (article [9966]). The output consists for the most part of texts dealing with other Italian artists closely associated with Giotto ([9523] Gaddi, [5474] Cimabue, [18357] Pisano, and [15149] Masaccio). In addition to the full articles listed in Table 4, two text sections are also retrieved (identified by the *e* suffix), and one text paragraph (identified by the *p* suffix). Thus, instead of displaying the full 25-page text of the article on Painting [17535], only section 15 of that article is obtained.

Table 4. Illustration of mixed retrieval consisting of full texts, text sections and text paragraphs in answer to [9966] Giotto ([17535.c15] Painting/Medieval Painting/Giotto [9590.p7] Fresco/History [9894.c4] Ghirlandaio/Domenico di Tommaso Bigordi di Ghirlandaio)

Document number	Query similarity	Retrieved item	Retrieval of partial document
9966	1.00	Giotto	-
17535	0.61	Painting	[17535.c15]
9523	0.51	Gaddi, Taddeo	-
5474	0.48	Cimabue	-
18357	0.45	Pisano, Andrea	-
4778	0.38	Cavallini, Pietro	-
15149	0.37	Masaccio	-
9590	0.34	Fresco	[9590.p7]
19464	0.33	Renaissance Art and Architecture	-
14148	0.31	Lippi, Fra Filippo	-
7402	0.30	Domenico Veneziano	-
9894	0.30	Ghirlandaio	[9894.c4]
17301	0.28	Orcagna, Andrea	-
250761	0.28	[illegible]	-

This subsection, entitled "Medieval Painting," specifically deals with Giotto. Similarly, paragraph 7 of [9590] covers the history of the Fresco in which Giotto has significantly contributed. Finally, section 4 of article [9894] deals with the one member of the Ghirlandaio family (Domenico Ghirlandaio) who was associated with Giotto. (The full article [9894] also examines other members of the Ghirlandaio family who had no relations with Giotto)

The decomposition of long texts into text components provides an independent treatment of text segments, and offers the potential for very precise retrieval of only the most immediately relevant items. Somewhat related attempts have been made over the years to implement viable text passage retrieval systems [10, 17]. However practical, automatic methods had not so far been developed in this area.

Automatic Generation of Hypertext Products

The global-local text comparison methods described earlier are applicable not only to information retrieval but also to text structuring. In that case, comparisons are performed between pairs of texts, or text excerpts, and sufficiently similar texts are linked, allowing joint access to related text excerpts.

A network structure, known as a hypertext, is normally proposed for the representation of linked text structures. The nodes of the network normally represent text excerpts, and the branches between pairs of nodes identify various relationships between text segments. Certain text relationships are easily detected automatically for example, the hierarchical relations between full documents and text sections and paragraphs included in these documents. The text processing system outlined in this study is useful for a fully automatic generation of semantically based hypertext networks whose construction is normally assumed to require assistance by human subject experts [8].

Hypertext networks are useful for collection browsing, allowing the users freely to navigate from one

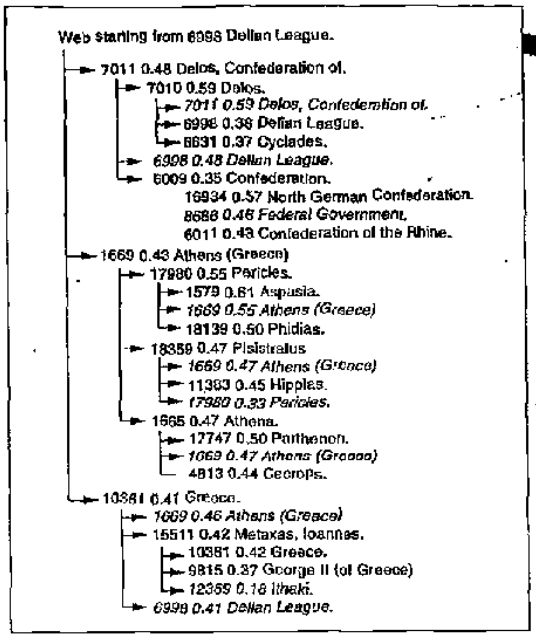


Figure 1. Linked hypertext web for document 6998 "Delian League" (depth 3–breadth 3 search; at least one matching sentence pair required at threshold 75.0; duplicated items are shown in italics)

topic area to a neighboring one, and from particular documents to other *related items*. In addition, hypertext networks can also be used as components of normal information retrieval activities [2, 4, 6, 15]. The retrieval output produced by conventional retrieval activities can then be modified in various ways by using the relations specified by the hypertext organization. New items related in the hypertext, for example, may be added to the retrieved set of items, or the output ranking obtained in a standard search may be modified by using the links between items contained in the hypertext.

Various kinds of structured text representations can be generated automatically as shown in the exam-

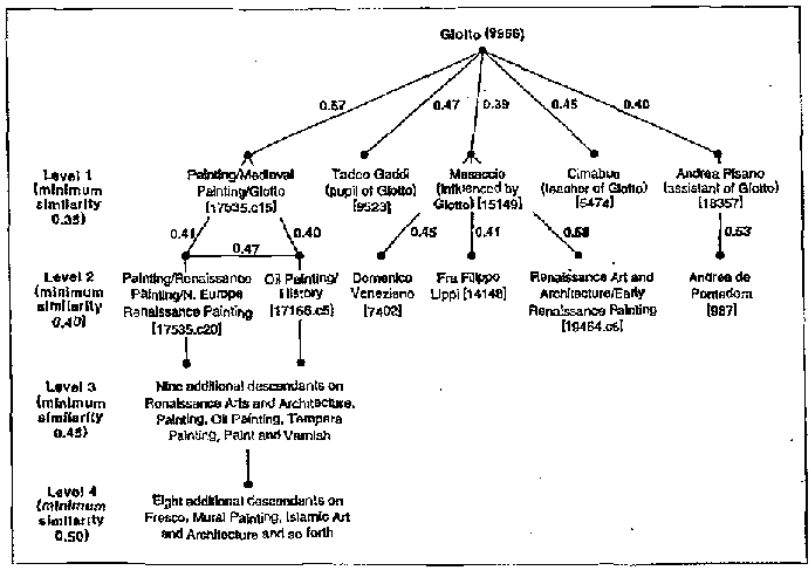
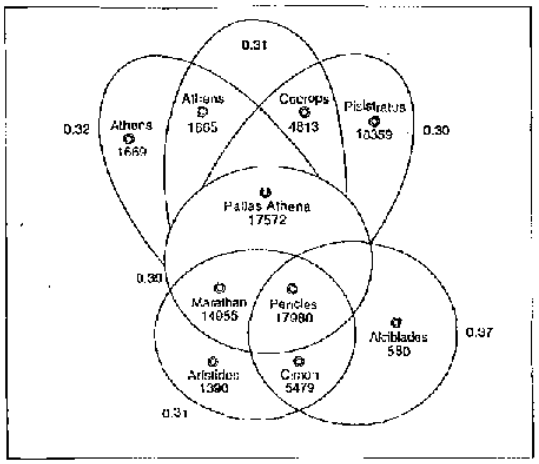


Figure 2. Self-contained map of items related to article 9956 "Giotto"



a. Complete link clusters including document 17980 "Pericles"

Document Number	Similarity	Title
6998	1.00	Delian League
1667	0.66	Athenian League
7011	0.48	Delos, Confederation of
1668	0.43	Athens
13779	0.43	League of nations
10381	0.41	Greece,
17572	0.40	Pallas Athena
17980	0.39	Pericles
7010	0.38	Delos
1104	0.35	Antaicidas
1390	0.35	Aristides
14955	0.34	Marathon
5479	0.33	Cimon
1665	0.33	Athena
23226	0.33	Urban League

0.31 0.39 0.32

b. Unrestricted search output for query 6998 "Delian League" with some identified cluster structures

Figure 3. Search results with identified cluster structures

ples of Figures 1 through 3. The results of a "breadth n-depth n" search, known as a web, are displayed in Figure 1. In this case, a base text is used for an initial search ([6998] Delian League, in the example of Figure 1), and the top n retrieved items are displayed on level 1 of the web. Each of these n items is then used as a new query and the n closest items to each of these are shown on level 2 of the web. This operation is repeated for n stages in all. The web of Figure 1 is the result of a breadth-3-depth-3 search. Subtrees generated several times are listed only once in the web of Figure 1.

Because many different search queries are used to produce the structured web output, the retrieved items become more and more removed from the original search topic as one proceeds from one tree level of the next. Thus, in the example of Figure 1, only the top-most subtree derived from [7011] Confederation of Delos, covers topics related to the Delian League.

In the illustration of Figure 2, the same number of items are retrieved in each search (3 for the sample web of Figure 1). A variable retrieval strategy is presented in Figure 2. In that example, the search starts with base document [9966] Giotto, and all retrieved items with a global query similarity exceeding 0.35 are listed on level 1 of the graph. Each of these items is then used for a new search, *using a higher retrieval threshold* (0.40) for level 2 of the tree. The process is continued for subsequent search levels with increased retrieval thresholds on each level. A self-contained map of related texts is produced by this process because the higher retrieval thresholds needed on the lower search levels will eventually fail to be met by any of the documents in the collection. The web of Figure 2 ends on level 4, because the search fails to locate additional items with a similarity to level 4 items exceeding 0.50.

A last example of text structuring involves the use of a clustering process, where mutually similar text items are grouped into affinity classes, or clusters. In the hypertext context, aggregates of similar nodes

have been introduced as a way of simplifying overly complex structures by creating supernodes, each of which represents a whole set of highly similar nodes. Methods have also been suggested for utilizing clustering systems in hypertext environments [1, 3, 5].

One simple method for a cluster-based text-linking and retrieval system consists in carrying out search and retrieval operations normally, while modifying the ranked output to take into account the available clustering information. Figure 3a contains a set of complete-link clusters, all of which include document [17980] Pericles. The overall clustering similarity, representing the minimum similarity between any pair of documents included in a common cluster, appears next to each defined cluster. Thus, the cluster similarity for the three-element cluster that includes [17980] Pericles, [14955] Marathon, and [17572] Pallas Athena is a relatively high 0.99 (on a scale from 0 to 1). This indicates that these three items are closely related.

The list of retrieved items obtained by an unrestricted search with query [6998] Defiant League appears in Figure 3b, with added clustering information. Assuming that a clustering threshold of 0.30 is considered significant, the retrieval of item [1669] Athens might immediately lead to four additional items grouped with Athens in a common cluster, including [17572] Pallas Athens, [17980] Pericles, [14955] Marathon, and [1669] Athena. In a cluster-based search strategy, these items could be retrieved ahead of the unclustered items that immediately follow Athens on the original retrieval list.

The use of clustering strategies in information retrieval environments has been limited in practice, because of the expense of generating the cluster structures for large collections of documents. In the proposed implementation, a global clustering operation becomes unnecessary. Instead, the clustering operation is applied locally to items jointly retrieved in response to available queries. In such circumstances, the clusters can be generated efficiently even when very large text files are in use.

Evaluation

The evaluation of information retrieval or text linking operations is a major unsolved problem. Normally, the effectiveness of text processing operations is evaluated by computing parameters such as recall and precision, defined as the proportion of relevant items retrieved, and the proportion of retrieved items that are relevant, respectively. The concept of document relevance must be settled outside the retrieval environment. If texts are used both as search queries and as retrievable items in response to queries, then relevance assessments, or judgments of related texts, are needed for all pairs of texts. The encyclopedia used in the current experiments contains over 200 million pairs of articles, and over one billion pairs of text sections. Obviously, full relevance-data can never be obtained for collections of that size, and hence complete recall-precision figures are not computable.

However, in the automated encyclopedia environment, fragmentary, objective relevance assessments are available if the cross-references built into the text are used as relevance indicators. In particular, if article A carries a cross-reference to article B (for example, "Lee Harvey Oswald," see "John F. Kennedy"), then one can postulate that article B is relevant when A is submitted as a query (but not vice-versa). In the *Funk and Wagnalls New Encyclopedia*, the available cross-reference structure is sparse,

Table 5. Evaluation of local sentence and paragraph matching (averages for 559 search requests)

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Unrestricted search	5228	586	0.8138	0.1106	— —
1 para 30	5212	596	0.8388	0.1188	+ 5.2%
1 para 40	5131	599	0.8379	0.1209	+ 7.1%
1 para 50	4939	601	0.8407	0.1283	+ 10.1%
1 para 75	4342	596	0.8148	0.1681	+ 12.5%
1 para 100	3949	580	0.7883	0.6079	+ 11.3%
1 para 125	3678	569	0.7764	0.2347	+ 11.5%
1 para 150	3512	546	0.7593	0.2401	+ 8.8%
1 para 200	3334	510	0.7268	0.2293	+ 1.9%
1 sent 30	4993	603	0.8368	0.1263	+ 7.1%
1 sent 40	4769	596	0.8260	0.1351	+ 7.5%
1 sent 50	4536	598	0.8250	0.1706	+ 13.4%
1 sent 75	3563	556	0.7619	0.2508	+ 10.8%
1 sent 100	2952	468	0.6569	0.2204	− 4.8%
1 sent 125	2416	417	0.5965	0.2241	− 18.1%
1 sent 150	1840	358	0.5230	0.2268	− 25.7%
1 sent 200	1069	228	0.3590	0.2332	− 45.3%

consisting of only 80,000 references between article pairs, out of the over 300 million that are theoretically possible. This implies that very few relevant articles are objectively identified with respect to each query article. Articles not identified as relevant must be treated as nonrelevant, making it impossible to compute reasonable values for search precision and recall.

Tentative performance information is, however, obtainable by using relative performance differences between different search runs (instead of absolute performance measurements) to judge the effectiveness of the retrieval methodologies. Table 5 contains such data for eight restricted searches involving both local paragraph comparisons, and local sentence matches. Specifically, the last column of Table 5 reflects the average improvement obtainable with the local paragraph and sentence-matching restrictions at the indicated local similarity thresholds over a base run represented by an unrestricted search using a global vector comparison only. The overall evaluation parameter used in the last column of the table consists of percentage *improvements* (or deteriorations) in the average search precision for 359 encyclopedia searches, evaluated at three recall points (0.20, 0.50, and 0.80). This measure represents a hybrid, single-value parameter that includes aspects of both recall and

Table 6. Examples of sentence comparison problems.

Document 1	Document 2
[7480] A.G. Dove	[16005] Moon
Sentence [7480.s5]	Sentence [16005.s21]
In his later career, Dove painted . . . the *Full Moon (1937, Phillips Collection)* in which the landscape is transformed into a cyclops with the *moon* for an eye.	About a week later, the *moon* is in first-quarter resembling a luminous half-circle; another week later, the *full moon* shows its lighted surface . . .
Global document similarity: 0.2198 Sentence pair similarity: 150.72 Common vocabulary: full (6.7%), moon (93.3%)	
a. **Inappropriate sentence match due to one highly-weighted common term ("moon")**	
[9562] Gall	[23008] J.M.W. Turner
Sentence [9562.s10]	Sentence [23008.s22]
The *gall* wasps, a family of small insects, *include* the greatest number of species of gall-forming insects.	Other famous works of this period *include* paintings of the *National Gallery (London)* and the National Gallery of Art (Washington)
Global document similarity: 0.2438 Sentence pair similarity: 98.59 Common vocabulary: gall (98.5%), include (1.5%)	
b. **Inappropriate sentence match due to one highly weighted truncated term ("gall," "gallery")**	
[9074] American football	[9615] Game theory
Sentence [9074.s150]	Sentence [9615.s34]
College *teams* usually play about 10 *games*, and professional teams play 16 *games* in the regular season, plus play-off *games* before the start of the season.	Two-person, or dual, *games* include the largest category of familiar *games* such as chess and checkers . . . or two-*team* games such as bridge; more complex conflicts *include* n-person *games* such as poker . . .
Global document similarity: 0.2452 Sentence pair similarity: 459.69 Common vocabulary: game (79%), play (3%), team (18%)	
c. **Inappropriate sentence match due to related but not identical topic ("American football," "game theory")**	

precision, and it has previously been used for global evaluation purposes [13].

Table 5 shows that small improvements in overall performance are obtainable even with very low paragraph or sentence-matching thresholds of 30 or 40. Matching local structures (sentences or paragraphs) will be present at such a low threshold for most retrieved articles, including many of the nonrelevant ones. As the local matching threshold grows, the advantage of the local processing grows, reaching a maximum at thresholds of 75.0 to 100.0 for matching paragraph restrictions, and 50.0 to 75.0 for matching sentence restrictions. As the local matching requirement becomes even stricter, some of the relevant retrieved items are no longer able to meet the local similarity requirement, and consequently fall by the wayside. This depresses the recall and affects the overall evaluation parameter. This becomes especially obvious for the sentence comparisons in which the performance decreases rapidly for sentence similarity thresholds greater than 100.0.

Average recall and precision values are also included in Table 5 to assess the performance after the retrieval of 15 documents per query. For some queries, it was impossible to find as many as 15 items with nonzero query similarity. In that case, a variable retrieval cut-off was used equal to the total number of encyclopedia articles exhibiting a nonzero similarity with the query. The recall values after 15 retrieved articles are generally high, indicating that the formally identified relevant articles are indeed retrieved among the top 15 items for most queries. The precision values are artificially low because most retrieved items are (falsely) declared as nonrelevant: formally only about 3 relevant articles exist on average for each query, but 15 are retrieved in each case.

The effect of the local search restrictions is most easily judged by considering the data in columns 2 and 3 of Table 5, giving the total number of retrieved items for the 359 search requests, and the total number of relevant retrieved items, respectively. As the table shows, the total number of retrieved items for all queries drops rapidly as increasingly severe local paragraph or sentence restrictions are applied. For example, when at least one matching paragraph pair is required above similarity threshold 75.0, only 4,342 documents are retrieved in all, compared with 5,218 for the unrestricted case—a decrease of 17%. However, the total number of relevant articles retrieved in that case does not decrease at all, but actually increases slightly from 588 to 596.

The same effect is noticeable for all restricted runs with a moderate local match requirement: the total number of items able to satisfy the local restrictions is much lower, and a few of the items ejected in the restricted run are replaced by other relevant items not previously retrieved. This accounts for the search precision being able to increase substantially for the restricted runs while the recall improves moderately or stays the same. Properly applied search restrictions then act principally as precision devices, where they serve to reject large numbers of nonrelevant items.

It was mentioned earlier that an informal, subjective examination of the items retrieved in answer to random search requests produces a search precision of about 95%. That is, only about 5% of the retrieved items appear to be marginal or inappropriate. Some problem cases that may arise in the sentence-matching process are presented in Table 6. By far the greatest number of inappropriate sentence matches is due to matching highly weighted terms that are used in different senses in the corresponding sentences. Such a term is "moon" in documents [7480] A.G. Dove (a painter) and [16005] Moon (see Table 6a). Over 99% of the sentence-matching coefficient is due to the term "moon" in that example. The difficulties presented by highly weighted single terms are compounded by the term truncation process, which forces matches between distinct words exhibiting the same word stem. Table 6b shows an example in which "gall" matches "galley" after truncation, forcing a document match between [9552] Gall and [25008] J.M.W. Turner (a painter). The truncated term "gall" accounts for over 98% of the sentence-matching coefficient in the example of Table 6b.

The sentence-matching problems caused by highly weighted, ambiguous single terms can be reduced substantially by using a complex sentence-matching requirement stipulating that the sentence pair similarity must exceed 75.0, and that in addition no single term should account for more than 90% of the total sentence-matching coefficient. With this added term-weighting restriction, the examples of Tables 6a and 6b, would no longer hurt the retrieval performance because the sentence-matching criterion could not be satisfied with this added restriction, and the offending documents would not be restricted. The single-term weight restriction unfortunately does not help with the example of Table 6c. In that case, related, but not identical topics are discussed using similar vocabularies ("game," "play," and "team" in the example of Table 6c). Without a deeper semantic analysis, distinct, but related document pairs such as those of Table 6c are difficult to distinguish.

Conclusion

A text manipulation system is introduced in this study, which uses global and local text-matching operations to identify stored texts likely to be of interest to system users. The text analysis and retrieval strategies described here are applicable to large collections of texts without subject matter restrictions, and to heterogeneous documents that vary widely in scope and extent. No large preconstructed knowledge sources are needed. However such tools can be added whenever they become available for use in unrestricted topic environments.

Retrieval strategies are described that make use of hierarchical text decomposition to successively refine the query statements and hence the coverage of the retrieved items. Additionally, several types of automatically constructed linked text structures are introduced. Such hypertext

products should make it possible to carry out efficient browsing operations among sets of related items, leading to efficient collection access by the user population.

Acknowledgment

The writers are grateful to the Microsoft Corporation for making available an electronic version of the *Funk and Wagnalls New Encyclopedia* for experimental purposes. The first author is also greatly indebted to the Alexander von Humboldt Foundation for supporting a research stay in Germany, and to Rainer Kuhlen and the information science group at the University of Konstanz for providing the attractive working environment where this study was performed.

References

1. Botafogo, R.A. and Shneiderman, B. Identifying aggregates in hypertext structures. In *Proceedings of Hypertext 91*, ACM, New York, Dec. 1991, pp. 63–74.
2. Clitherow, P., Riecken, D. and Muller, M. VISAR: A System for inference and navigation in hypertext. In *Proceedings of Hypertext 89*, ACM, New York, Nov. 89, pp. 293–304.
3. Croft, W.B. and Thompson, R.H. I3R: A new approach to the design of document retrieval systems. *J. Am. Soc. Inf. Sci. 33*, 6 (1987), 389–404.
4. Croft, W.B. and Turtle, H. A retrieval model incorporating hypertext links. In *Proceedings of Hypertext 89*. ACM, New York, 1989, 213–223.
5. Crouch, D.B., Crouch, C.J. and Andreas, G. The use of cluster hierarchies in hypertext information retrieval. In *Proceedings of Hypertext 89*, ACM, New York, Nov. 1989, pp. 225–237.
6. Fuhr, N. Hypertext and information retrieval. In P. Gloor and N. Streitz, Eds., *Hypertext and Hypermedia*, Springer Verlag, Berlin, 1990, pp. 101–111.
7. *Funk and Wagnalls New Encyclopedia*. Funk and Wagnalls, New York, 1979, 29 volumes.
8. Furuta, R., Plaisant, C. and Shneiderman, B. Automatically transforming regularly structured linear documents into hypertext. *Electronic Pub. 2*, 4 (Dec. 1989), 211–229.
9. Grice, H.P. *Studies in the Way of Words*. Harvard University Press, Cambridge Mass., 1989.
10. O'Connor, J. Answer passage retrieval by text searching. *J. Am. Soc. Inf. Sci. 32*, 4 (July 1980), 227–239.
11. Rocchio, J.J. *Relevance Feedback in Information Retrieval*, in *The Smart System—Experiments in Automatic Document Processing*, G. Salton, Ed., Prentice-Hall, Englewood Cliffs, N.J., 1971, 313–323.
12. Salton, G. Ed., *The Smart Retrieval System—Experiments in Automatic Document Processing*. Prentice Hall, Englewood Cliffs, N.J., 1971.
13. Salton, G. *Automatic Text Processing—The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley, Reading, Mass., 1989.
14. Salton, G. Developments in automatic text retrieval. *Science 253:5023*, (Aug. 30, 1991), 974–980.
15. Salton, G. and Allan, J. Selective text utilization and text traversal. In *Proceedings of Hypertext '91*. ACM, New York, 1991, pp. 131–144.
16. Salton, G., Allan, J. and Buckley, C. Selective use of full-text databases. Tech. Rep. TR 92-1300, Dept. of Computer Science, Cornell University, Ithaca, New York, Aug. 1992.
17. Salton, G., Allan, J. and Buckley, C. Approaches to passage retrieval in full-text information systems. In *Proceedings of the Sixteenth ACM/SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, 1993, pp. 49–58.
18. Salton, G. and Buckley, C. Term weighting approaches in automatic text retrieval. *Inf. Proc. Manage 24*, 5 (1988), 513–523.
19. Salton, G. and Buckley, C. Improving retrieval performance by relevance feedback. *J. Am. Soc. Inf. Sci. 41*, 4 (1990), 288–297.
20. Salton, G. and Buckley, C. Automatic text structuring and retrieval—Experiments in automatic encyclopedia searching. In *Proceedings of the Fourteenth ACM/SIGIR Conference on Research and Development in Information Retrieval*, ACM, New York, 1991, pp. 21–30.
21. Salton, G. and Buckley, C. Global text matching for information retrieval. *Science 253:5023*, (Aug. 30, 1991), 1012–1015.
22. Salton, G., Fox, E.A. and Wu, H. Extended Boolean information retrieval. *Commun. ACM 26*, 12 (Dec. 1983), 1022–1036.
23. Salton, G., Yang, C.S. and Wong, A. A vector space model for automatic indexing. *Commun. ACM 18*, 11 (Nov. 1975), 613–620.
24. Stanfill, C. and Kahle, B. Parallel free-text search on the Connection Machine. *Commun. ACM 29*, 12 (Dec. 1986), 1229–1239.
25. Wittgenstein, L. *Philosophical Investigations*. Basil Blackwell and Mott Ltd., Oxford, England, 1953.

CR Categories and Subject Descriptors: H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing; H.3.2 [Information Storage and Retrieval]: Record Classification; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—*evaluation/methodology*; I.2.7 [Artificial Intelligence]: Natural language processing

General Terms: Algorithms, design, experimentation, performance, theory

Additional Key Words and Phrases: Content analysis, full-text searching, information retrieval, text structuring

About the Authors:

GERARD SALTON has been a professor of computer science at Cornell University since 1965. He directs the Smart information retrieval project focusing on information retrieval and hypertext construction for large full-text document collections. email: gs@cs.cornell.edu

JAMES ALLAN is a research assistant in the computer science department at Cornell University. email: allan@cs.cornell.edu

CHRIS BUCKLEY is a research assistant in the computer science department at Cornell University. He wrote the most recent version (number 11) of the Smart text retrieval system. email: chrisb@cs.cornell.edu

Authors' Present Address: Department of Computer Science, Cornell University, Upson Hall, Ithaca, New York, 14853.

This study was supported in part by the National Science Foundation under grant IRI 89-15847

A distributed approach to high-performance information retrieval

Kenneth Baclawski* and J. Elliott Smith
Northeastern University
College of Computer Science
Boston, Massachusetts 02115
(617) 373-4631
FAX: (617) 373-5121
{kenb,esmith}@ccs.neu.edu

March 24, 1994

Abstract

A distributed architecture and indexing algorithm for high-performance information retrieval has been developed. A prototype system has been built that achieves a throughput of 500 queries per second with a response time of less than one second on an 8-node network of workstations. The algorithm is robust in the presence of lost and duplicated messages as well as failures of nodes of the network. The architecture can be scaled up to larger networks and higher levels of service. The retrieval model allows for semantically rich queries and information objects.

1 Introduction and Architecture

In this article, we discuss a distributed approach to high-performance information retrieval, called KEYNET. More details about the model and its architecture have been presented elsewhere [BS94a, BS94b, BS94c]. In this note, we sketch some background material about

*This material is based upon work supported by the National Science Foundation under Grant No. IRI-9117030.

KEYNET, and then discuss those aspects of the distributed algorithm that have not been presented elsewhere.

The KEYNET system is designed for information retrieval (IR) from a corpus of information objects in a single subject area. It is especially well suited for non-textual information objects, for example, scientific data files, satellite images and videotapes, although some kinds of textual document, such as research papers in a single discipline, can also be supported. The information objects may be physically located anywhere in the network. Retrieval is accomplished by means of a *content label* for each information object. These content labels are stored in a repository at the KEYNET site. The structure of the content labels is specified by an information model or *ontology*. The ontology can be as simple as a collection of document attributes such as "Author," "Title," "Publication Date," and so on, or it can be as complex as a general semantic network as used in artificial intelligence[Lehr92]. The content labels are indexed by means of a distributed hash table stored in the main memories of a collection of processors at the KEYNET site. These processors form the *search engine*.

To see more precisely where all of these components reside, and how they are connected to one another, refer to Figure 1. The user's computer is in the upper left. A copy of the ontology is kept locally at the user site. As this will require from several hundred megabytes to a few gigabytes of memory, it would generally be stored on a CD-ROM. The ontology is also the basis for the user interface to the search engine. Queries must conform to the format specified by the ontology, and are sent over the network to a front-end processor at the KEYNET site. Responses are sent back over the network to the user's site, where they are presented to the user using the ontology. The prototype system uses a connectionless

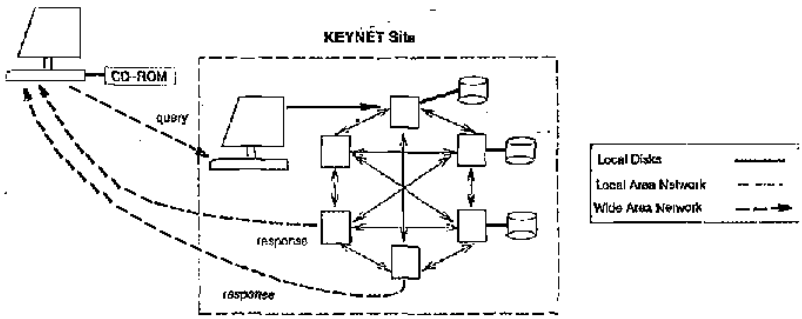


Figure 1: Architecture of KEYNET Search Engine

communication protocol so that no connection is required for making a query, and also so that the responses need not be sent back from the same computer that originally received the query.

At the KEYNET site, the front-end computer is responsible for relaying query requests to one of the search engine computers. The reason for having a front-end computer is mainly for distributing the workload but it also helps to simplify the protocol for making queries. The search engine itself is a collection of processors (or more precisely server processes) joined by a high-speed local area network. The search engine processors will be called *nodes*. The repository of content labels is distributed on disks attached to some of the nodes. The index to the content labels is distributed among the main memories of the nodes.

Since a connectionless communication protocol is unreliable, it is necessary for the user computer to resend the query if there is no response after a timeout period. The keynet protocol is stateless and idempotent, and so it works well with a connectionless communication

service.

2 Distributed Algorithm

The basic indexing strategy is to match probes (fragments of queries) with index terms (fragments of content labels). We now discuss the details of the distributed algorithm that accomplishes this matching. This algorithm can be characterized as a "scatter-gather" technique. Queries are sent to a front-end processor in the form of datagrams. The front-end processor assigns a query id, acknowledges receipt of the query and forwards the query to a randomly chosen node of the search engine. This is the first scattering step. The node that is assigned the query is called the *home node* of the query.

At the home node, the query is broken apart into probe fragments. For the details of the fragmentation algorithm, see [Bac94]. For the purposes of this article, one can regard the fragments as small pieces of the original query. These fragments overlap one another, especially in semantically complex queries.

Each probe is then hashed using a standard hashing algorithm. The hash value is in two parts. One part is a node number and the other part is the local hash value used at that node. The local hash value and the query identifier are then sent to the node that was selected by the hash value. This forms the second scatter step of the algorithm. The result of hashing is to scatter the probes uniformly to all of the nodes of the search engine.

Upon receiving the local hash value of a probe, the node looks it up in its local hash table. An index term in the hash table that matches a probe is called a *hit*. The hits are sent back to the home node of the query. This is the "gather" step of the algorithm. Special

trailer messages are used for determining when all the hits of all the probes of a query have been collected. The home node then computes the similarity measure (using an IR measure of similarity called the cosine measure) of each object in the collection, and the objects are ordered by the degree of similarity to the query. The object identifiers of the most relevant objects are then sent back to the user.

The insertion of a new content label in the index is done in a manner very similar to the query algorithm. Since content labels and queries are both keynets conforming to the same keynet ontology, they use exactly the same data structure. The same fragmentation, hashing and scattering algorithms are used for content labels as for queries. The only difference is that instead of matching entries in the hash table, index terms are inserted into the table. Note that index terms are not explicitly stored in the index, just their hash values are stored. The number of bits in the hash value is chosen to be so large that it is very unlikely that two probes would have the same hash value. As a result it suffices to store only a hash value and not the index term itself. Since an index term is nearly always much larger than a local hash value, this results in a significant savings of space with only a slight reduction in retrieval effectiveness.

The query algorithm presented so far represents the *basic* level of service. Higher levels of service are provided by using additional scatter-gather operations. The overall structure of the various levels of service is shown in Figure 2. The second level of service uses two scatter-gather operations. After completing the collection phase of the basic level of service, the home node sends each object identifier to the node where its content label is stored, resulting in yet another scatter step. The content label is then retrieved and sent back to

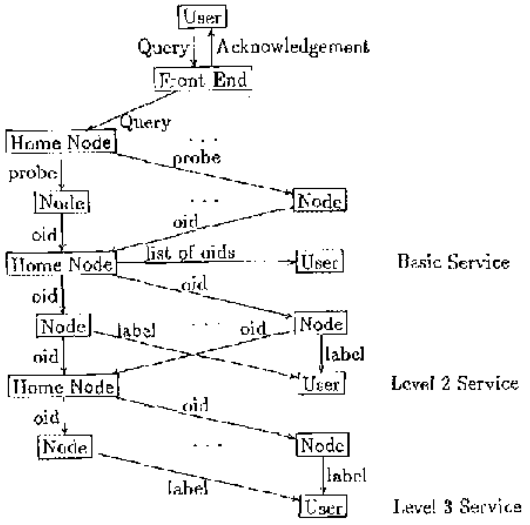


Figure 2: The KEYNET Distributed IR Algorithm

the user's computer where the content labels are gathered, arranged and displayed using the keynet ontology.

The highest level of service is level 3. Instead of sending the content labels back to the user as in level 2, the nodes perform a structural analysis of the content labels each of which is compared with the original query using subgraph isomorphism techniques. The result of this analysis is a new estimate of the degree of relevance of each information object with the original query. These new estimates are sent back to the home node which gathers them and constructs a new ranking of the object identifiers. In the process the least relevant objects will be dropped from the list. The object identifiers are then sent back to the nodes where the content labels reside so that the content labels can be sent to the user with their final ranks.

The KEYNET algorithm is a "shared nothing" algorithm: each processing node has responsibility for its own local memory (both main memory and disk storage) and there is no shared memory. This is in contrast with shared memory paradigms for parallel and distributed computation, such as the Linda model [CG89]. Nevertheless, the KEYNET does share some aspects with the Linda model. For example, communication is one-way and messages can be processed in a different order than they were sent. Furthermore, the messages of KEYNET are tuples, and while it is important that a message be processed at a particular node, it is not important which thread processes it at that node.

3 Reliability

An important characteristic of the KEYNET system is its reliability. Unlike many distributed algorithms which are sensitive to the possibility that information might be lost, the KEYNET algorithm can tolerate the loss and duplication of packets as well as the failure of one of the nodes of the search engine. Such occurrences may cause some degradation in retrieval effectiveness, but they do not stop query processing. The advantage of being able to tolerate communication failures is that a connectionless communication protocol can be used. Such a protocol is unreliable but has better performance than a connection-oriented protocol.

We give some examples to illustrate how the algorithm can tolerate communication and node failures. Suppose the user's original query packet were lost. The user's computer would fail to receive the acknowledgement, and it would time out and send the query again. If the acknowledgement were lost, then the user's computer would send the query packet again. The effect is exactly the same as the duplication of the query packet. Since KEYNET is a stateless, idempotent server, there is no harm in requesting the same query twice, Since responses will be labeled with the query identifier, the user's computer would be able to detect and to discard the spurious responses.

Within the KEYNET search engine, loss or duplication of packets is rare since communication is over a local area network, but they can still occur. Loss of a probe effectively deletes the part of the query that is represented by the probe. Because the fragments of a query overlap one another, the probes are redundant. The role played by the lost or duplicated probe is therefore likely to be compensated by probes that overlap it. In a similar way, the

loss of a hit message may change the overall ordering of the information objects that are retrieved but since the retrieval of a document generally requires several hits for it to be considered sufficiently relevant, one can tolerate the loss of one hit.

The loss of a node is a more serious failure than just the loss of a message. It results in the loss of a large number of probes and hits. However, if there are many nodes, then the effect on any one query is relatively small on the average. So even this kind of failure can be tolerated in much the same way that a simple loss of a message can be tolerated.

4 Performance

We have developed a prototype KEYNET system that runs on a network of sparcstations connected by a local area network. The sparcstations are part of the network of Unix workstations maintained by the College for a large community of faculty, students, staff and guests. We ran our tests late at night on relatively unused workstations. There was less activity on the network at these times, but there was some activity even at 3:00 AM, so our results exhibit a variance that reflects this.

The largest search engine we have used consisted of an 8-node network. On each node we index the content labels of 20,000 information objects. The individual nodes are implemented as servers. Specifically, they are implemented as connectionless, multi-threaded, interrupt-driven, stateless servers. Each server is responsible for a fixed amount of memory, 16 Mbytes, which is small enough for page faulting to be rare so that, to a first approximation, the 16 Mbytes can be regarded as physical memory.

Messages between nodes are buffered and sent in groups to improve throughput at the

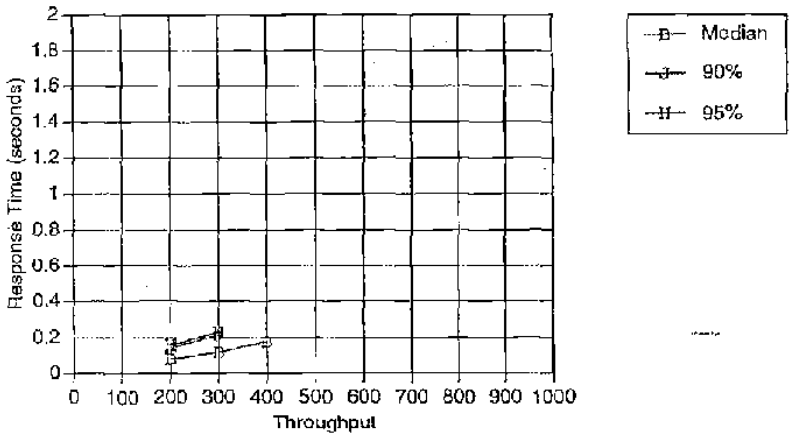


Figure 3: Response-Time versus Throughput on a Four-Node Engine

expense of some response time. The amount of buffering can be adjusted so as to maximize throughput for a given response time requirement. In the last runs, the buffer size was adjusted for each configuration so that the frequency with which packets are being sent is approximately the same. Otherwise, when one varies the number of nodes, all one is measuring is the effect of the buffer size. We have a mechanism for flushing buffers when this is deemed to be appropriate. Load balancing is done by measuring the relative performance of the nodes at the beginning of each run, and then allocating tasks to the nodes using this measurement.

In figures 3 and 4, we show the median, 90^{th} and 95^{th} percentile response times versus throughput for 4- and 8-node search engines, respectively. The 4-node search engine has a better response-time than the 8-node search engine for lower values of the throughput, but

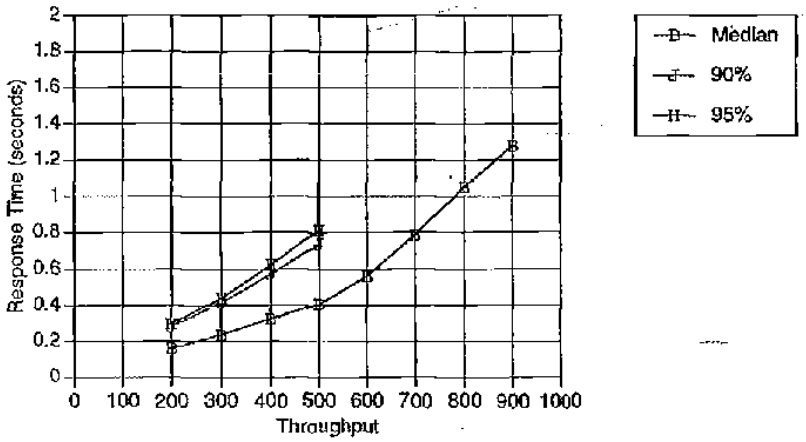


Figure 4: Response-Time versus Throughput on an Eight-Node Engine

for higher values of the throughput, the 4-node response-time gets so large that it goes off the scale. The 8-node engine can can sustain a much higher throughput before the response-time goes off the scale.

5 Summary

A distributed algorithm has been developed for high-performance information retrieval from a subject-specific corpus of information objects. A prototype system has been developed to measure the performance characteristics of the algorithm, and the prototype has achieved a throughput of 500 queries per second. The algorithm uses a local memory model and connectionless communication both of which are very useful for scaling up to larger corpora. The algorithm supports several levels of service and can tolerate a small number of

communication and processor failures.

References

[Bac94] K. Baclawski. An abstract model for semantically rich information retrieval. *Information Systems*, 1994. to be submitted.

[BS94a] K. Baclawski and J. E. Smith. High-performance, distributed information retrieval. Technical Report NU-CCS-94-05, Northeastern University, College of Computer Science, 1994.

[BS94b] K. Baclawski and J. E. Smith. KEYNET: Fast indexing for semantically rich information retrieval. Technical Report NU-CCS-94-06, Northeastern University, College of Computer Science, 1994.

[BS94c] K. Baclawski and J. E. Smith. A unified approach to high-performance, vector-based information retrieval. In *RIAO'94*, 1994. submitted.

[CG89] N. Carriero and D. Gelernter. Linda in context. *Comm. ACM*, 32:444–458, 1989.

[Leh92] Fritz Lehmann. Semantic networks in artificial intelligence, parts I and II. *Computers and Mathematics with Applications*, 23(2-9), 1992.

A unified approach to high-performance, vector-based information retrieval

Kenneth Baclawski* and J. Elliott Smith
Northeastern University
College of Computer Science
Boston, Massachusetts 02115
(617) 373-4631
FAX: (617) 373-5121
{kenb,esmith}@ccs.neu.edu

March 21, 1994

Abstract

An information retrieval model based on the vector space model is proposed that unifies and extends many commonly used retrieval mechanisms. A distributed architecture and indexing algorithm for high-performance retrieval using this model has been developed. A prototype system has been built that achieves a throughput of 500 queries per second with a response time of less than one second on an 8-node network of workstations. The model and algorithm are designed for retrieval from a corpus of information objects in a single subject area. The objects need not be textual, and must be annotated with content labels. With current technology, our system can be scaled up to support a corpus of several million information objects. Finally, the model allows for content labels that are semantically more complex than just attributes, keywords and subject classifications.

1 Introduction

With the expansion of the Internet and the development of new "Information Superhighways," computer-based communication is becoming the defining technology of this decade. The amount of information that will be available over these new networks is immense: on

*This material is based upon work supported by the National Science Foundation under Grant No. IRI-9117030.

the order of billions of objects and hundreds of terabytes of data. Information retrieval in such an environment is a monumental task but essential to its success. We propose an information retrieval model, called KEYNET, that unifies and extends many commonly used IR mechanisms. We have also developed a distributed architecture and indexing algorithm for high-performance IR using the KEYNET model. Our prototype system has achieved a throughput of 500 queries per second with a response time of less than a second for more than 95% of the queries. This measurement was done locally and therefore does not include any wide area network delay times.

The KEYNET system is designed for IR from a corpus of information objects in a single subject area. It is especially well suited for non-textual information objects, for example, scientific data files, satellite images and videotapes, although some kinds of textual document, such as research papers in a single discipline, can also be supported. With current technology, KEYNET can support very high-performance IR from a corpus having up to several million information objects at approximately the same level of performance as smaller corpora.

We begin by presenting the architecture of the KEYNET system. This will explain where the various kinds of information are located, the pathways for communication, and how a user interacts with a KEYNET search engine. In section 3 we introduce the KEYNET model and explain how it can be used to implement many commonly used mechanisms of information retrieval. We then turn to the details of the indexing algorithm. The algorithm is based on the vector space model for information retrieval. It differs from the usual vector space IR systems in using distributed hash tables rather than trees for indexing. The algorithm is presented in section 4. The prototype and its performance, and in particular how it scales up, are discussed in section 5. Although the KEYNET system can be used solely to implement one or more traditional IR mechanisms, it can also support semantically richer content labeling of information objects. Some examples are presented in section 6 to illustrate this feature of KEYNET. We discuss related work in section 7, and we conclude with a summary and future work planned for KEYNET in the last section.

2 The KEYNET Architecture

The purpose of KEYNET is to assist in retrieving information objects from a corpus of them. These information objects need not be textual and may be physically located anywhere in the network. Retrieval is accomplished by means of a *content label* for each information object. These content labels are stored in a repository at the KEYNET site. The structure of the content labels is specified by an information model or *ontology*. The content labels are indexed by means of a distributed hash table stored in the main memories of a collection of processors at the KEYNET site. These processors form the *search engine*. Each content label contains information about locating and acquiring the information object. The KEYNET system is only concerned with finding information objects; users are responsible for actually acquiring (and presumably paying for) information objects.

To see more precisely where all of these components reside, and how they are connected to one another, refer to Figure 1. The user's computer is in the upper left. A copy of the ontology is kept locally at the user site. As this will require from several hundred megabytes to a few gigabytes of memory, it would generally be stored on a CD-ROM. The ontology is also the basis for the user interface to the search engine (see section 6). Queries must conform to the format specified by the ontology, and are sent over the network to a front-end processor at the KEYNET site. Responses are sent back over the network to the user's site, where they are presented to the user using the ontology. The prototype system uses a connectionless communication protocol so that no connection is required for making a query, and also so that the responses need not be sent back from the same computer that originally received the query.

At the KEYNET site, the front-end computer is responsible for relaying query requests to one of the search engine computers. The reason for having a front-end computer is mainly for distributing the workload but it also helps to simplify the protocol for making queries. The search engine itself is a collection of processors (or more precisely server processes) joined by a high-speed local area network. The search engine processors will be called *nodes*. The repository of content labels is distributed on disks attached to some of the nodes. The index

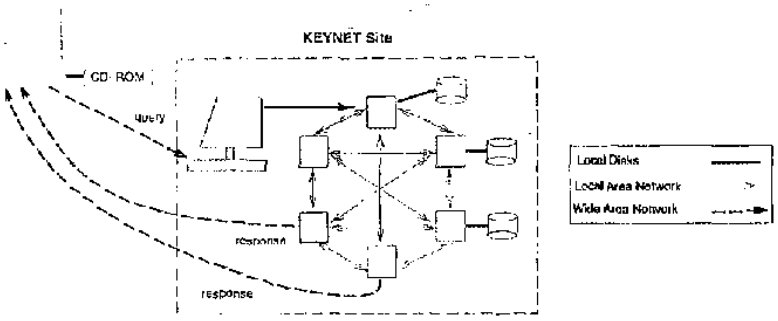


Figure 1: Architecture of KEYNET Search Engine

to the content labels is distributed among the main memories of the nodes. The prototype *differs from* the KEYNET architecture only in that it randomly generates the repository as well as queries sent to it.

Since a connectionless communication protocol is unreliable, it is necessary for the user computer to resend the query if there is no response after a timeout period. The keynet protocol is stateless and idempotent, and so it works well with a connectionless communication service. There is a similar protocol for registering information objects by sending content labels to the KEYNET site, but this is not explicitly shown in Figure 1.

3 The KEYNET Model

Both queries and content labels are represented using a data structure called a *keynet*. Intuitively, a keynet is semantically intermediate between a keyword and a semantic network. We build up the formal definition of a keynet in stages. The mathematical basis for the KEYNET model is first presented. Then the notion of a keynet ontology is defined, and it is used as the basis for defining the concept of a keynet. The keynet structure is used as the data structure for content labels, queries, index terms, etc. A more precise and complete mathematical formulation is given in [BS94].

The underlying mathematical structure on which keynets are based is the *directed graph*. See [CLR90, Section 5.4] for the basic definitions.[1] A directed graph consists of a set of vertices and a set of (directed) edges that link one vertex with another (possibly the same) vertex. Vertices and edges will generally have additional structure such as textual labels, informal explanations, etc. Edges, in particular, will be labeled with *type information*. When drawing a directed graph, one uses boxes for vertices and solid arrows for edges, each of which is labeled with some of the additional structure belonging to it. The *dimension* of a directed graph is the number of edges it has. A directed graph is *connected* if every pair of vertices can be linked by a sequence of edges (not necessarily all going in the same direction).

As we mentioned earlier, a KEYNET system requires a subject-specific knowledge model or ontology. The word ontology literally means "a branch of metaphysics relating to the nature and relations of being." Our use of the word is much more restrictive, dealing only with the nature of, and relationships among, concepts within a narrow subject area. Attempts to specify ontologies for scientific disciplines are very common, with most disciplines having some kind of subject classification scheme by this time.

The KEYNET system depends on having a background ontology that defines the structure and behavior of keynets. A *keynet ontology* consists of three parts:

1. A directed graph called the *schema*.

   (a) The vertices can be regarded as the set of *conceptual categories* of the ontology. They consist primarily of the subject classification categories for the subject covered by the corpus. However, a vertex can also be a property or attribute of an information object, such as an author, its year of publication, etc.

   (b) The links of the schema join conceptual categories. They are grouped into *link types*. The links in one link type share a common semantics. The best known example of a link type is the ISA link type, which is used to define the concept hierarchy for a subject classification. Other link types include "part of," "elabo-

[1] The only difference between our concept of directed graph and the standard one is that we allow more than one directed edge from one vertex to another.

ration of," "cause of," etc.

2. A set of terms or concepts, called the *lexicon* or *thesaurus*. One think of the terms either as lexical terms, and hence the *keywords* of the ontology, or on higher semantic level as concepts (each of which can be expressed in many ways, by words or phrases). While a lexical term is often a word or phrase, it can also be a person, a number, or a range of names or numbers.

3. A many-to many relation from the lexicon to the set of conceptual categories that specifies which category (or categories) a lexical term specializes or instantiates. Each lexical term is required to be an instance of at least one conceptual category.

The Unified Medical Language System (UMLS) [HL93] of the National Library of Medicine is an example of a subject-specific ontology. The UMLS ontology is more complex than a keynet ontology; for example, it supports lexical relationships like synonymy as well as relationships on the concept level. However, the core of the UMLS ontology can be regarded as a keynet ontology.

The reason for splitting concepts into the schema and lexicon levels is pragmatic. While there will generally be only about 100 conceptual categories and even fewer link types, there will typically be on the order of 100,000 lexical terms. By defining links only at the schema level, where there are fewer conceptual categories to deal with, it is easier to understand the ontology.

One commonly used IR mechanism is the use of attributes to identify documents. For example, the names of authors or words from the title. These are represented in the ontology with "Author" and "Title" concept categories. The lexicon would have the names of authors and (key)words from titles (or ranges of these if the number of authors or title words was too large). Ranges can also be used to express "wild card" matches.

The notion of a keynet ontology is the basis for the concept of a keynet. A keynet *conforming to a keynet ontology consists of the following:*

1. A directed graph.

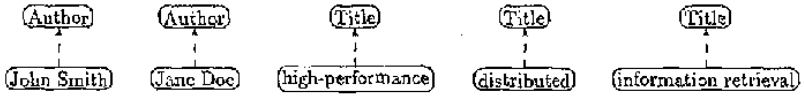
(a) The vertices of the directed graph are copies of vertices of the ontology. There can be several copies in a keynet of one vertex in the ontology.

(b) The edges of the directed graph are copies of edges in the ontology. Each edge of the keynet must link copies of the source and destination vertices of the corresponding edge in the ontology: the keynet must faithfully reflect the corresponding structure in the ontology.

2. A set of lexical terms that are copies of lexical terms in the ontology.

3. A *one-to-one* function from lexical terms to vertices. This function specifies the keynet vertex that is being instantiated by each lexical term. Unlike the ontology where the relationship between lexical terms and categories is many-to-many, each category vertex in a keynet may be instantiated at most once by a lexical term, and every lexical term instantiates exactly one vertex.

For example, the following keynet would be used for a content label of the document "High-performance, distributed information retrieval," by John Smith and Jane Doe:



The examples given so far have not made use of any edges to link conceptual categories. In Figure 2, there is an example of a keynet conforming to the UMLS ontology. This

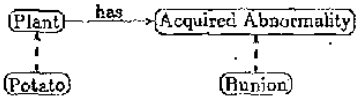
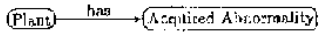


Figure 2: Can a potato get bunions?

keynet can be expressed in English in several ways. Its expression as a query using English automatically generated using stock phrases is "Find all documents in which the plant potato

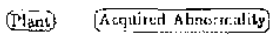
has an acquired abnormality bunion." Its expression as a statement in a content label is "The potato can exhibit an abnormality functionally similar to bunions." Finally a more *succinct colloquial expression* would be: "Can potatoes get bunions?" While this may seem to be a somewhat whimsical query, it makes perfectly good sense, and we will later discuss in section 6 how a KEYNET system would understand and respond to such a query.

Keynets are used not only for content labels and queries but also for index terms. However, we do not use arbitrary keynets as index terms because that would result in a "combinatorial explosion" of possibilities. Rather we restrict attention to a special kind of keynet. A *fragment* of a keynet is a connected subset of the keynet. In other words, a fragment consists of a choice of vertices, edges and lexical terms from a keynet such that the set of vertices and edges so chosen forms a connected directed graph. For example, this is a fragment of the keynet in Figure 2:



and here is another:

However, the keynet



is not a fragment because it is not connected.

The main result for fragments is the fact that the number of fragments having small dimension grows linearly with the size of the keynet from which they are taken.

Theorem 1 *Let K be a keynet having v vertices. If the degree of a vertex of K is at most d, then there are at most $2+6(v-1)+4dv$ fragments of K having at most 2 edges.*

For a proof of the theorem, see [BS94]. The bound in the theorem is actually a worst case bound, and in practice one does much better, typically just 4 times the number of vertices of the keynet. A fragment having exactly one edge is called a *clasp*, while a fragment having exactly two edges is called a *double clasp*.

The reason for limiting fragments to be at most double clasps is not just pragmatic. It arises from a well-known result from Cognitive Science known as the 7 ± 2 rule[Mil56]. This rule states that the maximum number of conceptual "chunks" that can be *manipulated* by a person at one time is in the range 5 to 9 chunks. A clasp consists of 3 to 5 parts (the two vertices, the edge and up to two instances of the vertices), and a double clasp consists of 5 to 8 parts. Therefore limiting to just one vertex or a single clasp would be too limiting, while a double clasp almost exactly matches the 7 ± 2 rule.

Intuitively, retrieval in the KEYNET system proceeds by matching fragments of a query (called *probes*) with fragments of content labels (called *index terms* or *index fragments*). The degree of relevance is then determined by a similarity measure between the vector of probes and the vector of index terms. For this to scale up to large corpora, the number of index terms must not be too large. In particular, if we allowed arbitrary subsets of a content label to be index terms, then the number of index terms would grow exponentially in the size of the content label. The theorem assures us that the size of the index will be manageable.

4 Indexing Strategy

As we have already mentioned, the basic indexing strategy is to match probes (fragments of queries) with index terms (fragments of content labels). We now discuss the details of the distributed algorithm that accomplishes this matching. This algorithm can be characterized as a "scatter-gather" technique. Queries are sent to a front-end processor in the form of datagrams. The front-end processor assigns a query id, acknowledges receipt of the query and forwards the query to a randomly chosen node of the search engine. This is the first scattering step. The node that is assigned the query is called the *home node* of the query.

At the home node, the query is broken apart into probe fragments as discussed in section 3 above. The fragmentation algorithm is more subtle than one would expect, since loops and multiple edges are allowed in keynets. The details and proof of this algorithm are given in [BS94]. Each probe is then hashed using a standard algorithm given in [Knu73, Section 6.4]. The hash value is in two parts. One part is a node number and the other part is the local

hash value used at that node. The local hash value and the query identifier are then sent to the node that was selected by the hash value. This forms the second scatter step of the algorithm. The result of hashing is to scatter the probes uniformly to all of the nodes of the search engine.

Upon receiving the local hash value of a probe, the node looks it up in its local hash table. The hash table algorithm we employ is called "open addressing with double hashing," as described in [Knu73, Section 6.4]. We found that this technique is very space efficient and that collisions do not affect performance even when the hash table is 90% full. An index term in the hash table that matches a probe is called a "hit." The hits are sent back to the home node of the query. This is the "gather" step of the algorithm. Special trailer messages are used for determining when all the hits of all the probes of a query have been collected. The home node then computes the similarity measure (currently the cosine measure) of each object in the collection, and the objects are ordered by the degree of similarity. The object identifiers of the most relevant objects are then sent back to the user.

The insertion of a new content label in the index is done in a manner very similar to the query algorithm. Since content labels and queries are both keynets conforming to the same keynet ontology, they use exactly the same data structure. The same fragmentation, hashing and scattering algorithms are used for content labels as for queries. The only difference is that instead of matching entries in the hash table, index terms are inserted into the table. Note that index terms are not explicitly stored in the index, just their hash values are stored. The number of bits in the hash value is chosen to be so large that it is very unlikely that two probes would have the same hash value. As a result it suffices to store only a hash value and not the index term itself. Since an index term is nearly always much larger than a local hash value, this results in a significant savings of space with only a slight reduction in retrieval effectiveness.

The query algorithm presented so far represents the basic level of service. Higher levels of service can be provided by using additional scatter-gather operations. For example, the second level of service uses two scatter-gather operations. After completing the collection phase of the basic level of service, the home node sends each object identifier to the node

where its content label is stored, resulting in yet another scatter step. The content label is then retrieved and sent back to the user's computer where the content labels are gathered, arranged and displayed using the keynet ontology. Still higher levels of service are discussed in [BS94].

5 Performance

We have developed a prototype KEYNET system that runs on a network of sparcstations connected by a local area network. The sparcstations are part of the network of Unix workstations maintained by the College for a large community of faculty, students, staff and guests. We ran our tests late at night on relatively unused workstations. There was less activity on the network at these times, but there was some activity even at 3:00 AM, so our results exhibit a variance that reflects this.

The largest search engine we have used consisted of an 8 node network. On each node we index the content labels of 20,000 information objects, using 16 Mbytes of memory. Although the Sparcstations have anywhere from 64 to 128 Mbytes of memory, we found that attempting to allocate more than 16 Mbytes resulted in excessive paging activity. Presumably this is a result of the other activities, some of them in the background, going on at all times throughout the network.

The prototype is fully distributed, using a pure message-passing communication mechanism. All messages are one-way: no process ever waits for a reply to a message. The memory model is local, i.e., a "shared nothing" system. It is not a parallel processing algorithm, but we are investigating whether it can be ported to a parallel machine architecture.

The individual nodes are implemented as servers. Specifically, they are implemented as connectionless, multi-threaded, interrupt-driven, stateless servers. Each server is responsible for a fixed amount of memory, 16 Mbytes, which is small enough for page faulting to be rare so that, to a first approximation, the 16 Mbytes can be regarded as physical memory.

Messages between nodes are buffered and sent in groups to improve throughput at the expense of some response time. The amount of buffering can be adjusted so as to maximize

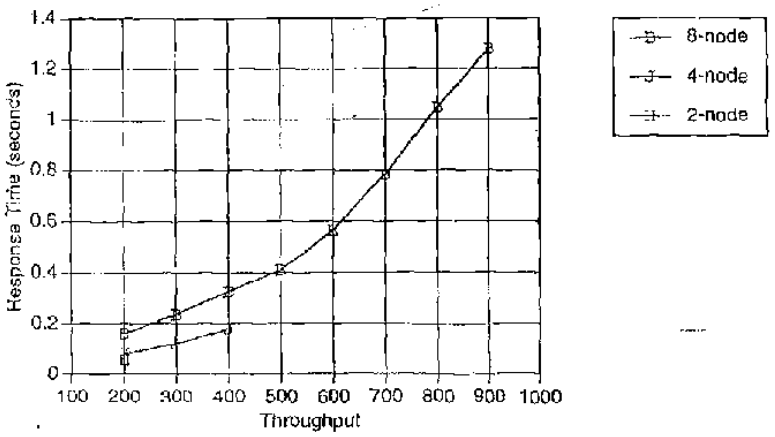


Figure 3: Median Response-Time versus Throughput

throughput for a given response time requirement. In the test runs, the buffer size was adjusted for each configuration so that the frequency with which packets are being sent is approximately the same. Otherwise, when one varies the number of nodes, all one is measuring is the effect of the buffer size. We have a mechanism for flushing buffers when this is deemed to be appropriate. Load balancing is done by measuring the relative performance of the nodes at the beginning of each run, and then allocating tasks to the nodes using this measurement.

In figures 3 and 4, we show the median and 95th percentile response times, respectively, versus throughput for 2-, 4- and 8 node search engines. The 2-node engine has the best response time for 200 queries/second, but for a larger throughput its response time goes off the scale. Increasing the number of nodes to 4 results in a slightly slower response at 200 queries per second, but now the throughput can be increased to 400 queries per second before the response time goes off the scale. The 8-node engine has the slowest response at low throughputs, but can sustain the highest throughput before the response time goes off

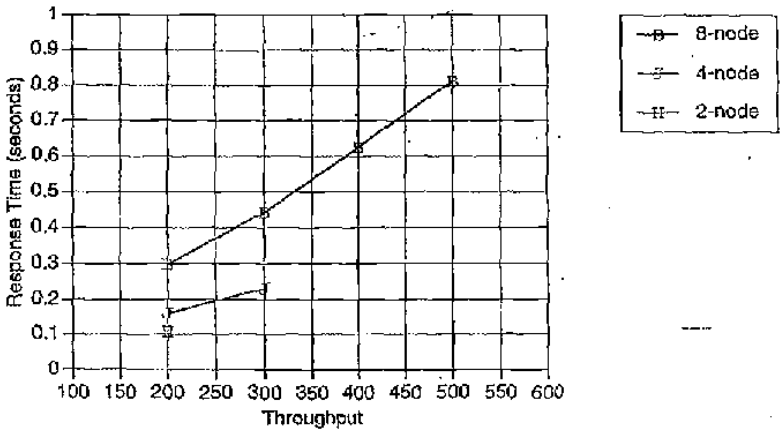


Figure 4: 95th Percentile of Response-Time versus Throughput

the scale.

The prototype is running well enough to obtain throughput and response-time data on randomly generated databases and queries, and the user interface understands UMLS. However, these two components were not yet integrated at the time this paper was written.

6 Semantically Rich Information Retrieval

In addition to the commonly used IR mechanisms, KEYNET supports a semantically richer form of information retrieval. However, as Lakoff points out[Lak87], "Human categorization is *based on principles that extend far beyond those envisioned* in the classical theory." As a result, simple classification methods leading to taxonomies of concepts are inadequate for expressing the rich variety of human categorization techniques.

Unfortunately, since no IR systems currently use such a mechanism, one cannot evaluate whether it would improve retrieval effectiveness. Our contribution is a "proof-of-existence" that there are no technological or user interface barriers to building a high-performance IR

system of this kind.

KEYNET gains potential semantic leverage relative to traditional vector space methods by responding to relations between keywords, in addition to (*possibly accidental*) *co-occurrences*. Consider the query "Can potatoes get bunions?" of Figure 2. In this case, the entire query represents one of the fragments. Other fragments may be regarded as varying degrees of generalization of the original query. For example, the specific concept 'potato' may be abstracted into the conceptual category 'plant' to yield the fragment "plants that have bunions;" or 'bunion' may be replaced by the conceptual category 'acquired abnormality' to produce the corresponding fragment "potatoes that have acquired abnormalities."

The best matches for the query "Can potatoes get bunions?" will, roughly speaking, be the content labels that include the largest number of index terms (content label fragments) that match the query fragments. Therefore an information object that deals with plants that get bunions or potatoes with acquired abnormalities is considered a better match than an information object that discusses, say, potatoes as a treatment for bunions, or mentions potatoes and also bunions but in different contexts. This is the sense in which a KEYNET system can be said to have "understood" a query.

7 Related Work

While semantic networks from AI were influential in the development of the KEYNET model, its indexing technique is fundamentally based on the vector space model for IR [Sal89]. From the point of view of IR, KEYNET can be regarded as a mechanism for unifying a number of different IR mechanisms, and the KEYNET system can be regarded as a *high-performance*, distributed search engine that can be applied effectively to vector-based retrieval from a corpus of annotated documents. From this point of view, keynets serve the same role as subject classification categories, keywords and properties such as author, title or date of publication.

Despite their reputation in IR circles as cumbersome, inefficient and suitable only for small databases, at least one IR researcher has used knowledge-based indices successfully [FHL+91].

Fuhr *et al*'s AIR/X performs automatic indexing of documents using terms (descriptors) from a restricted vocabulary. Probabilistic classification determines indexing weights for each descriptor using rule-based inference. KEYNET differs from AIR/X in using a form of semantic network as part of the retrieval algorithm rather than in the extraction of suitable terms to be used for indexing. The extraction of terms (or in our case *clasps*) *from a corpus of* textual information objects is an important problem. One of the projects related to KEYNET is an effort to automate the extraction of *keynets* from biological research papers, in particular the *Materials & Methods* sections of such papers. See [BFFP93, BFH+93a, BFH+93b]. However, such extraction is independent of the architecture and algorithm employed for retrieval (and isn't even possible for non-textual information objects).

After building a system similar to AIR/X, Jacobs [Jac93] determined that "the combination of statistical analysis and natural language based categorization is considerably better than either alone." His paper describes an automated set of statistical methods for pattern acquisition that operate inside a knowledge-based approach for news categorization (an area closely related to document classification and other IR tasks). Like AIR/X, Jacobs' system does not employ semantic networks in the IR engine. Another difference between KEYNET and Jacobs' system is that KEYNET is designed for a corpus of documents in a single subject area, where it is feasible to develop a subject-specific ontology. Developing an ontology for heterogeneous textual documents is a formidable task, many orders of magnitude larger than is feasible with current technology.

The EDS TemplateFiller system [SMHC93] applies Message Understanding (MUC) text-filtering techniques to the generation of knowledge frames for one or a few specific subject areas from entire texts (computer product announcements). TemplateFiller fills in slots for frames that exist in a predefined schema of templates, ignoring subjects that are not in the schema.

There are many other MUC-style systems; in fact, there is an annual competition among them. Such a system can automate the construction of the content labels for a collection of specialized textual documents. In a project related to the KEYNET project, we are building a MUC-style system for biological research papers [BFH+93a].

A structural model of IR was developed by a project at the University of Western Ontario[Lu90]. This model uses case relations as the structure. *Case relations* are a *major* component of case grammars which are a tool proposed by linguistic theorists and developed by computational linguists for natural language processing. The term "case" here is a refinement and generalization of well-known grammatical relationships such as "subject," "object" and "indirect object." Similarity of a query to a document is measured using a form of structural similarity. One conclusion of the study was that the proposed IR mechanism does not improve retrieval effectiveness. Although this system has some superficial similarity to KEYNET it differs in a number of important respects. The most important difference is that the Western Ontario system uses "surface" syntactic structures while KEYNET uses conceptual structures. Another important difference is that the Western Ontario system combines the two tasks of knowledge extraction from text with IR of the resulting knowledge structures. The first task is known to be very difficult, with the best such systems (the MUC-style systems discussed above) achieving only about 50% accuracy, and even this requires that the documents be restricted to a specialized topic area. Accuracy is much lower when general documents are being analyzed.

Several families of databases for semantic networks have been developed. Such databases are often called knowledge-base systems. Some of the best known of these are: Conceptual Dependency, ECO, KL-ONE, NETL, Preference Semantics, PSN and SNePs (see [Lch92]). All of these support link types, frame systems and so on, but few if any explicitly concern themselves with performance measures familiar to work in IR. Hence it is not surprising that these techniques have acquired a reputation for being cumbersome, inefficient and suitable only for small databases. The KEYNET system shows that it is possible to use a limited form of semantic network model in a high-performance IR system.

Some other examples of knowledge-based query modification systems include systems primarily for information retrieval such as those in [GS93, CD90, Har92, QF93] as well as systems designed for database systems such as the KNOWIT system of Sølvberg, Nordbø and Aamodt [SNA92] and the cooperative query answering system in [CC92]. All of these are front-end query modification systems added to an IR or database system. Such query

modification techniques can also be used with a KEYNET system; in fact, a keynet ontology is very well suited to the support of such techniques; and the high-performance of a KEYNET search engine is useful for supporting the much larger queries generated by query modification techniques. However, such techniques are fundamentally a front-end for the actual search engine.

8 Summary and Future Work

An IR model has been introduced that generalizes many commonly used IR mechanisms. A distributed architecture and indexing algorithm have been developed for this model. A prototype system has been developed to measure the performance characteristics of the algorithm, and the prototype has achieved a throughput of 500 queries per second. The model can support semantically complex content labels and queries.

A number of research efforts are actively being pursued on both KEYNET and closely related projects.

- Since a KEYNET ontology is required for indexing a corpus using a KEYNET search engine, it is important to have good tools for building such an ontology. The ontology is also closely related to the user interface to the KEYNET system, making the ontology even more fundamental. The OntologyBuilder[BF93] is a research effort related to KEYNET that addresses the problem of building and managing ontologies. Even a small ontology can occupy hundreds of megabytes of storage, so database management is clearly an important part of this problem. The OntologyBuilder uses object-oriented database management tools for managing the very large and complex data structures of an ontology.

- The indexing strategy used by KEYNET is a distributed hashing method. One feature it does not yet have is expandability. Several distributed, expandable hashing methods have been proposed, and we plan to integrate KEYNET with one of them.

References

[BF93] K. Baclawski and N. Fridman. M&M-Query: Database support for the annotation and retrieval of biological research articles. Technical Report NU-CCS-94-07, Northeastern University, College of Computer Science, 1993.

[BFFP93] K. Baclawski, R. Futrelle, N. Fridman, and M. Pescitelli. Database techniques for biological materials & methods. In *First Intern. Conf. Intell. Sys. Molecular Biology*, pages 21–28, 1993.

[BFH+93a] K. Baclawski, R. Futrelle, C. Hafner, M. Pescitelli, N. Fridman, B. Li, and C. Zou. Data/knowledge bases for biological papers and techniques. In *Proc. Sympos. Adv. Data Management for the Scientist and Engineer*, pages 23–28, 1993.

[BFH+93b] K. Baclawski, R. Futrelle, C. Hafner, M. Pescitelli, N. Fridman, B. Li, and C. Zou. M&M-Query: Materials & Methods knowledge base and query system. Technical Report NU-CCS-93-06, Northeastern University, College of Computer Science, 1993.

[BS94] K. Baclawski and J. E. Smith. An abstract model for semantically rich information retrieval. *Information Systems*, 1994. to be submitted.

[CC92] Wesley W. Chu and Qiming Chen. Neighborhood and associative query answering. *Journal of Intelligent Information Systems*, 1(3/4):355–386, December 1992.

[CD90] H. Chen and V. Dhar. Online query refinement on information retrieval systems: A process model of searcher/system interactions. In *Proc. SIGIR*, pages 115–133, 1990.

[CLR90] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.

[FHL+91] Norbert Fuhr, Stephan Hartmann, Gerhard Lustig, Michael Schwantner, Konstadinos Tzeras, and Gerhard Knorz. AIR/X: A Rule-Based Multistage Indexing System for Large Subject Fields. In *Proc. User-Oriented Content-Based Text and Image Handling Conference (RIAO-91)*, pages 606–623, Centre de Hautes Etudes Internationales d'Informatique Documentaire, Paris, France, 1991.

[GS93] S. Gauch and J. Smith. An expert system for automatic query reformulation. In *J. Amer. Soc. Info. Sys.*, volume 44, pages 124–136, 1993.

[Har92] D. Harman. Relevance feedback revisited. In *Proc. SIGIR*, pages 1–10, 1992.

[HL93] B. Humphreys and D. Lindberg. The UMLS project: making the conceptual connection between users and the information they need. *Bulletin of the Medical Library Association*, 81(2):170–177, 1993.

[Jac93] Paul S. Jacobs. Using statistical methods to improve knowledge-based news categorization. *IEEE Expert*, pages 13–23, April 1993.

[Knu73] Donald Knuth. *Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, Reading, MA, 1973.

[Lak87] George Lakoff. *Women, Fire and Dangerous Things*. The University of Chicago Press, Chicago, IL, 1987.

[Leh92] Fritz Lehmann. Semantic networks in artificial intelligence, parts I and II. *Computers and Mathematics with Applications*, 23(2-9), 1992.

[Lu90] X. Lu. *An Application of Case Relations to Document Retrieval*. PhD thesis, The University of Western Ontario, 1990. ISBN: 0-315-59092-0.

[Mil56] G. Miller. The magical number 7±2: some limits on our capacity for information processing. *Psych. Rev.*, 63:81–97, 1956.

[QF93] Y. Qiu and H. Frei. Concept based query expansion. In *Proc. SIGIR*, pages 160–169, Pittsburgh, PA, 1993.

[Sal89] Gerard Salton. *Automatic Text Processing*. Addison-Wesley, Reading, MA, 1989.

[SMHC93] H. Kelly Shuldberg, Melissa Macpherson, Pete Humphrey, and Jamil Corley. Distilling Information from Text: The EDS TemplateFiller System. *Journal of the American Society for Information Science*, 44(9):493–507, 1993.

[SNA92] Ingeborg Sølvberg, Inge Nordbø, and Agnar Aamodt. Knowledge-based information retrieval. *Future Generation Computer Systems*, 7:379–390, 1991/1992.

KEYNET: An architecture and protocol for high-performance semantically rich information retrieval

Kenneth Baclawski* and J. Elliott Smith
Northeastern University
College of Computer Science
Boston, Massachusetts 02115
(617) 373-4631
FAX: (617) 373-5121
{kenb,esmith}@ccs.neu.edu

April 29, 1994

Abstract

We propose an architecture and protocol for semantically rich information retrieval from a subject-specific corpus of information objects. The technique assumes that information objects have been labeled using small directed graphs, called content labels. A content label subsumes the role of an abstract. Both content labels and queries are required to conform with a general framework (ontology) for the subject area. Our method avoids explicit isomorphism testing by decomposing queries into fragments of small maximum size and using tables of these fragments to accomplish comparisons. The index structure is compatible with distributed processing architectures and scales up well, allowing very large collections to be searched very quickly using semantically complex queries.

1 Introduction.

With the expansion of the Internet and the development of new "Information Highways," computer-based communication is becoming the defining technology of this decade. A num-

*This material is based upon work supported by the National Science Foundation under Grant No. IRI-9117030.

ber of proposals have been made to build a coherent structure over these new high-bandwidth networks and thereby convert them into a National Information Infrastructure (NII). The amount of information that will be available in an NII is immense: on the order of billions of objects and hundreds of terabytes of data. Information Retrieval (IR) in such an environment is a monumental task but essential to the success of the infrastructure. Any solution to this problem must satisfy two important requirements: it must be scalable to handle the large number of information objects and it must be semantically rich enough to support effective information retrieval.

Our architecture and indexing strategy provides a search engine that can satisfy these requirements. The KEYNET system supports information retrieval for a corpus of information objects in a single subject area, such as a collection of biological research articles, a set of court cases, files containing remote geophysical sensor data, or even collections of software programs and modules. KEYNET unifies and extends many commonly used IR mechanisms, and can be used effectively not only for corpora consisting of annotated information objects, but also for object-oriented databases in general. A distributed architecture and indexing algorithm has been developed for high-performance IR using the KEYNET model[BS94c, BS94b]. The prototype system has achieved a throughput of 500 queries per second with a response time of less than a second for more than 95% of the queries[BS94d].

The KEYNET system is designed for IR from a corpus of information objects in a single subject area. It is especially well suited for non-textual information objects, such as scientific data files, satellite images and videotapes. For example, the literal content of a satellite image does not include the geographic coordinates of the boundaries of the image or other

cartographic abstractions. Some kinds of textual document, such as research papers in a single discipline, can also be supported. With current technology, KEYNET can support very high-performance IR from a corpus having up to several million information objects at *approximately the same level of performance as* smaller corpora.

A KEYNET system requires the development of a *subject-specific concept ontology* that is understandable to a literate practitioner of the field. A keynet ontology represents knowledge using a directed graph of conceptual categories and relationships between them. The Unified Medical Language System (UMLS) developed by the National Library of Medicine is an example of such an ontology[HL93, LHM93]. KEYNET further assumes that each information object in its collection has been annotated with a content label that indicates what portion of the subject-specific ontology relates to the content of the object. [1]

Both content labels and queries have the same data structure called the *keynet* structure. A keynet may be regarded as a kind of semantic network[Lev92], although in practice it is semantically intermediate between keywords and semantic networks. The keynet framework generalizes many commonly used mechanisms for information retrieval, such as: subject classification schemes, keywords, document abstracts, reviews, content labels for non-textual information objects, properties such as author or date of publication, ranges of text strings such as "wild card" match strings, and ranges of quantities. The KEYNET system allows a uniform treatment of these disparate techniques in a system that permits a great deal of flexibility compared to traditional database and information retrieval systems. For example,

[1] We are misusing the word "annotate" slightly here. Although a content label may refer to portions of its information object, its role is to classify or abstract rather than to explain the portions of the information object to which it refers.

one can combine all of the above mechanisms in a single system, and easily add new features to the ontology, such as new attributes and keywords. In addition, the KEYNET framework allows for sequences of concepts linked by relationships and expressed in natural language using phrases, clauses, sentences and paragraphs.

A content label is similar to an abstract or review of a document both in size and in being separately accessible from its corresponding information object. Using a tool such as our M&M-Query System[BF93], a content label can be generated by the author of the information object with no more effort than is now taken to write the abstract or to select the keywords.

2 Example

For a simple example of a content label consider an imaginary paper entitled, "POOQ: A parallel, object-oriented query system." Suppose that the paper uses some known dynamic programming algorithms to optimize queries for use on parallel machine architectures. A keyword-based approach could classify this paper by using phrases such as "parallel algorithms," "object-oriented databases," "dynamic programming," and "query optimization."

Keywords can include topically relevant words and phrases that do not appear in the information objects themselves. Furthermore, some information objects (like images, scientific data and software) have no text that can reasonably be used by traditional IR technology. Keynets enrich the semantics possible with keywords (simple subject classifications) by adding relationships between keywords. In addition to being more expressive than sets of keywords, keynets exhibit more structure and are generally larger, although still much

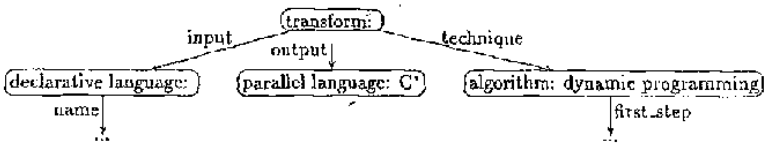


Figure 1: Example of part of a content label

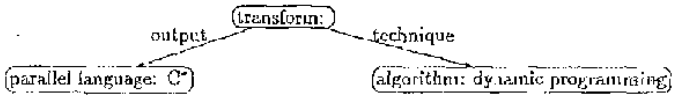


Figure 2: Semantic network of a query

smaller than the entire information object.

To describe a content label for the imaginary POOQ paper above, we must first describe a hypothetical ontology for Computer Science. Assume that one of the classes in this ontology is concerned with the concept of translation or transformation. The content label for the POOQ paper begins with an instance of the transformation class which is linked via an attribute edge labeled "input" to an object having the subtype "declarative language," as shown in Figure 1. The label of each node consists of its type followed by its value (if any) separated by a colon. Since the output language is well-known, no further elaboration is needed. However, the input language, POOQ, is not well-known, so it must be specified further with attributes like its name and other attributes not shown.

As an example of a query, consider "What systems use dynamic programming to generate C* code?" This is translated into the keynet in Figure 2.

After converting a query into a keynet, it is decomposed into components of bounded size. These fragments are called *probes*. The fragmentation algorithm is given in [BS94a]. The probes are indexed using a distributed hash index. Content labels are also fragmented and the fragments inserted into the index. The search step consists of hashing the probes and looking for matches with fragments of content labels. The *POOQ document in the example* has fragments that match every probe of the query.

3 Information Retrieval.

Information retrieval systems are primarily concerned with the problem of providing mechanisms for a user to select a small set of relevant documents (or parts of documents like chapters, figures, tables, and so on) from a large document collection. This objective differs from database management in a number of ways. Generally speaking, IR systems are not concerned with answering detailed queries about the content of the documents in the collection. On the other hand, database systems can answer detailed queries very precisely, but most are not capable of answering the vaguely worded queries about relevance that an IR system can handle.

A database system takes for granted that a query is precisely stated, and the issue becomes how efficiently the query can be evaluated. By contrast, since IR queries are not required to be precise, one measures performance in a different manner. The two most common general measures of retrieval quality in IR research are called *recall* and *precision*. The former is the ratio of documents retrieved versus the number of available documents relevant to the query, i.e., the fraction returned out of all desirable documents (a measure

of alpha risk or type I error). The latter is the ratio of the number of relevant documents retrieved versus the total number of documents retrieved, or the useful fraction of what was actually retrieved (beta risk or type II error).

Recall and precision presume that categories assigned by human experts are correct, complete, and well-specified; yet emergent concepts are seldom clearly formulated. An *ideal information* retrieval mechanism must not only capture poorly expressed concepts, but also somehow adapt as ideas change; its conceptual ontology must evolve over time.

Most commercial IR systems are based on a boolean model of relevance. The query terms are matched to keywords or to words in the document content, and relevance is determined by the satisfaction of a boolean expression specified by the user. A variety of techniques such as word stemming, truncation, thesauri and lexicons have been used to extend this model[Sal89].

In contrast to boolean methods, the so called "vector" methods use a notion of relevance that is less sharply defined: a document has a degree of relevance (salience) rather than simply being relevant or irrelevant. Documents are represented as points in a multidimensional vector space. One can then compare documents to each other as well as to queries. One commonly used measure is the cosine of the angle between the points regarded as vectors [Sal89]. Other *possible measures* of salience include path distance between nodes in a graph, or the number of levels that must be traversed to connect two categories in a hierarchy of abstractions.

While vector methods use probability and statistical methods to improve retrieval effectiveness, they are the same as boolean methods in their reliance on simple linguistic units,

such as combinations of words or phrases, as the basis for retrieval. Since fragments of natural language do not always communicate a concept unambiguously for every combination of speaker, listener, writer or reader, retrieval errors inevitably arise from irrelevant discourse. Consequently, to improve retrieval effectiveness IR systems often label documents using keywords or phrases that may never appear in the document itself.

Moreover, relevance requires relative judgement; material irrelevant for categorization may still be relevant for other user purposes. Retrieval that merely matches against text in a document body presumes concepts can be completely characterized by statistical correlations. Yet as Jacobs [Jac93] observes, "statistical methods must be an aid rather than a replacement for knowledge acquisition". Text statistics are best used to identify patterns that depend on specialized words and phrases not obvious to casual readers. Mechanisms that recognize complex ideas are better constructed by human experts, whose understanding of a concept may transcend language.

Unfortunately, knowledge-based approaches that utilize semantic networks are currently considered so inefficient that they are explicitly omitted from some IR textbooks. For example, [FBY92] dismisses them on the basis of "the amount of manual effort that would be needed to represent a large document collection."

4 Related Work.

Despite their reputation in IR circles as cumbersome, inefficient and suitable only for small databases, at least one IR researcher has used knowledge-based indices successfully [FHL+91]. Fuhr *et al*'s AIR/X performs automatic indexing of documents using terms (descriptors)

from a restricted vocabulary. Probabilistic classification determines indexing weights for each descriptor using rule-based inference.

After building a system similar to AIR/X, Jacobs [Jac93] determined that "the combination of statistical analysis and natural language based categorization is considerably better than either alone." His paper describes an automated set of statistical methods for pattern acquisition that operate inside a knowledge-based approach for news categorization (an area closely related to document classification and other information retrieval tasks).

Both of these systems attempt to automate the process of generating index terms. KEYNET, by contrast, is concerned not with how the index terms are obtained but instead with their structural relationships.

Several distinct families of databases for semantic networks have been developed. Such databases are often called knowledge-base systems. Some of the best known of these are: Conceptual Dependency, ECO, KL-ONE, NETL, Preference Semantics, PSN and SNePs (see [Leh92]). All of these support link types, frame systems and so on, but few if any explicitly concern themselves with performance measures familiar in traditional database work, such as minimizing the number of *disk accesses* required to retrieve complex structures (i.e., graphs assembled from multiple frames and their typed relations). Contemporary knowledge-base systems traverse semantic networks one frame/relation at a time. Unless the knowledge base fits entirely in the main memory of a single processor, these traversals result in large amounts of virtual memory paging.

In [Lev91], Levinson describes a technique for pattern-oriented (graph) retrieval. Levinson's approach discovers common structures among graphs in a database, and then uses

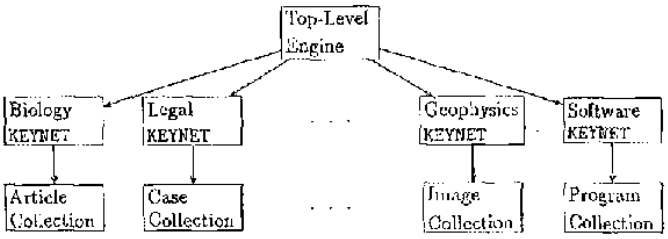
pattern associativity to reduce the required number of tests for subgraph isomorphism. A separate paper, [Lev92], compares techniques for subgraph isomorphism testing suitable for pattern-oriented retrieval. The baseline retrieval method described in [Lev92] considers a flat set of N networks with no pattern associativity information, in which each query requires N isomorphism tests. The first improvement indexes commonly occurring substructures in the graphs, and tests only a subset of the entire database. A second improvement creates a multilevel index of subgraph relationships between successive levels of substructures and the domain of graphs in the database (establishes a partial order). A final elaboration applies multilevel indexing to connectivity and label information required during subgraph isomorphism testing (via the refinement method), rather than to the graph substructures themselves. This produces a tree of "node descriptors" in order of increasing specificity, with actual index pointers at its leaves.

In KEYNET we *approximate* subgraph isomorphism testing to determine the relevance of information objects that were previously annotated with content labels. Our approach compiles fragments of content labels into a hash table, reducing lookup computation (while increasing database construction overhead). Although some ambiguity remains possible when matching fragments instead of whole graphs, vertex overlap between components that contain vertex-incident edges ensures similarity of graph structure between a query and a content label that is retrieved using its fragments. This approximation further allows "simultaneous" (multi-process, distributed or parallel) matching of different regions of the query and content labels (rather than serialized path-oriented inspection).

The KNOWIT system of Sølvberg, Nordbø and Aamodt [SNA92] embeds a semantic network

in a front-end query refinement system. Queries to the ESA-QUEST bibliographic database are expanded according to a semantic model that describes the meaning of a concept entirely in terms of its relations to other concepts in the model. The KNOWIT system is a front-end to a traditional IR system, whereas KEYNET uses semantic networks as part of its search strategy.

The Government Information Locator Service (GILS)[GIL] is an example of a second-level retrieval mechanism in which the result of a search *is another search engine rather* than an information object. The KEYNET model can also be applied to to instances of itself, producing a quasi-encyclopedic classification of information objects. The following diagram suggests how one could organize search engines in the NII:



Using currently available technology, each search engine could support collections having a few million information objects. The top-level search engine could, in turn, support one million search engines. The entire structure would therefore index 10^{12} information objects having an aggregate storage size of 10^{16} bytes, i.e., 10,000 Terabytes.

5 System Architecture.

The KEYNET information retrieval technique is designed to be used in a highly distributed environment, and assumes that the information objects themselves are widely distributed. Information objects need not be textual and may be physically located anywhere in the network.

Retrieval is accomplished by means of *content labels* for each information object. These content labels are stored in a repository at the KEYNET site. Structure that exists among the content labels is defined by a schema called the *ontology* that is a substantial database in its own right. For more details about its structure, see [BS94a]. The content labels are indexed by means of a distributed hash table stored in the main memories of a collection of processors at the KEYNET site. These processors form the *search engine*. Each content label contains information about locating and acquiring the actual information object. The KEYNET system is only concerned with finding information objects. Acquiring (and paying for) objects is a separate issue.

To see more precisely where all of these components reside, and how they are connected to one another, refer to Figure 3. The user's computer, responsible for presentation (user-interface) services, is at the upper left. A copy of the ontology is kept locally at the user site. As this will typically require several hundred megabytes of memory, it will generally be stored on a CD-ROM. The ontology is also the basis for the user interface to the search engine[BF93]. Queries must conform to structure specified by the ontology, and are sent over the network to a front-end processor at the KEYNET site. Responses are sent back over the

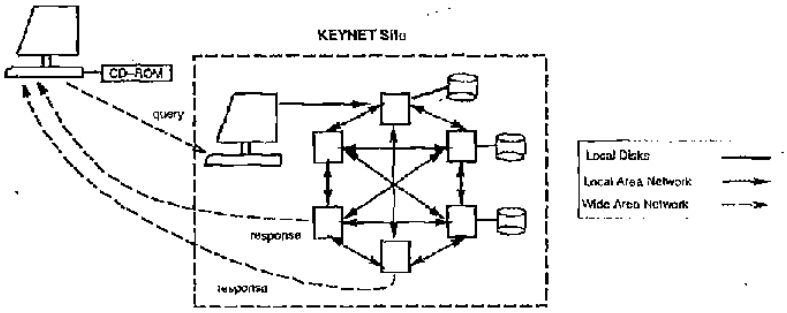


Figure 3: Architecture of KEYNET Search Engine

network to the user's site, where they are presented to the user (making use of the ontology to display them).

At the KEYNET site, *the front-end computer is responsible for relaying query requests to* one of the search engine computers. The purpose of the front-end computer is mainly to distribute the workload, but it also helps to simplify the protocol for making queries. The search engine itself is a collection of processors (or more precisely server processes) joined by a high-speed local area network. The search engine processors are called *nodes*. The repository of content labels is distributed on disks attached to some of the nodes. The index to the content labels is distributed among the main memories of the nodes.

Queries are answered by fragmenting them into small graph components, called *probes*, each having a bounded size. Components similarly obtained from content labels are called *index terms*.

The result of the fragmentation step is a large number of probes that can be indexed

in parallel. The indexing step is analogous to the technique used by biologists to study the genome. A chromosome (a very long strand of DNA) is probed using small pieces of DNA which can attach to the chromosome only where they match in a precise fashion. In fact, a KEYNET system can be used for the problem of mapping long strands of DNA called *clones* to their locations in the chromosome by regarding the clones as "information objects" which are "retrieved" using probes.

The basic indexing strategy of KEYNET is to match probes (fragments of queries) with index terms (fragments of content labels). We now give an outline of the distributed algorithm that accomplishes this matching. For more details see [BS94b]. This algorithm can be characterized as a "scatter-gather" technique. Queries are sent to a front-end processor that forwards the query to a randomly chosen node of the search engine. This is the first scattering step. The node that is assigned the query is called the *home node* of the query.

At the home node, the query is broken apart into probe fragments, as discussed earlier. Each probe is then hashed using a standard algorithm. The hash value is in two parts. One part is a node number and the other part is the local hash value used at that node. The local hash value and the query identifier are then sent to the node that was selected by the hash value. The result of hashing is to scatter the probes uniformly to all of the nodes of the search engine.

Upon receiving the local hash value of a probe, the node looks it up in its local hash table. An index term in the hash table that matches a probe is called a "hit." The hits are sent back to the home node of the query; this is the "gather" step of the algorithm. The home node then computes the similarity measure (currently the cosine measure) of each

object in the collection, and the objects are ordered by their degree of similarity. The object identifiers of the most relevant objects are then sent back to the user. More sophisticated techniques (including *explicit graph isomorphism testing*) can be applied as post-processing filters.

The insertion of a new content label in the index is done in a manner very similar to the query algorithm. The same fragmentation, hashing and scattering algorithms are used for *content* labels as for *queries*. The only difference is that instead of matching entries in the hash table, index terms are inserted into the table.

6 Formats and Protocol.

KEYNET uses the UDP protocol of the TCP/IP communications protocol suite. In this section we describe the format and protocol for KEYNET communication. Since KEYNET assumes that *the ontology* is available at both the sender and the receiver, it is unnecessary for the format to include any textual information, and *all fields consist of integers*. For convenience in communicating between different machine architectures, all integers are represented as *network integers*. For simplicity in the following, we have omitted the specification of acknowledgement and error structures.

6.1 Keynet Structure.

Both queries and content labels use the same data structure, called the keynet structure and defined as follows:

1. **Ontology Identifier.** Each ontology is assigned a unique integer identifier.

2. **Major version number.** Since ontologies can evolved over time, a version number is used to distinguish both major and minor versions of an ontology. Minor versions differ from one another only by the addition of new concepts, conceptual categories and relationship types. Major versions may differ in more substantial ways, including the *splitting of categories*, *merging* of categories, as well as more complex alterations in the ontology.

3. **Minor version number.**

4. **Count.** Most keynets are expected to be small enough to fit into a single UDP packet, but there is a mechanism for larger keynets. The count field specifies that the keynet has been split into a number of pieces as specified in this field. Normally the value of this field is 1.

5. **Sequence number.** When a keynet is in several pieces, this field will have the sequence number of this piece. The values range from 0 to one less than the count field above.

6. **Vertex count.** This is the count of the number of vertices that will be specified in the list immediately following this field.

7. **Vertex list.** This is a set of zero or more vertex specifications. Each vertex specification consists of three integers as follows:

   (a) **Vertex id.** Each vertex of a keynet has a unique identifier within the keynet.

(b) **Type id.** Each vertex has a type or conceptual category. The types and their identifiers are listed in the ontology.

(c) **Instance id.** A vertex may be instantiated using one of the instances given in the ontology. Instance identifiers are nonzero integers. If this field has value 0, then the vertex has not been instantiated.

8. **Edge count.** This is the count of the number of edges that will be specified in the list immediately following this field.

9. **Edge list.** This is a set of zero or more edge specifications. Each edge specification consists of three integers as follows:

   (a) **Source vertex.** This is the vertex identifier of the source of this edge in the keynet.

   (b) **Destination vertex.** This is the vertex identifier of the destination of this edge in the keynet.

   (c) **Edge type id.** Each edge has a type. The edge or relationship types and their identifiers are listed in the ontology.

6.2 Query Structure.

The query structure is used to send queries to a KEYNET search engine. This same structure is also used within the search engine for sending queries from one node to another. The query structure has the following structure:

1. **Message type specifier.** The first field is used to specify to a KEYNET search engine that this packet contains a query or a part of one.

2. **Source internet number.** This is the internet number of the internet node that originated the query. If this node sets this field to zero, then the engine will fill in its value.

3. **Source port number.** This is the port number used to send the query. Like the internet number, if this is set to zero by the originator, then the search engine will fill in its value.

4. **Source identifier.** This is a source-generated query identifier which may be used by the system that originates the query to distinguish its queries from each other. However, it does not distinguish queries originating from different locations.

5. **Engine identifier.** This is an identifier supplied by the search engine that uniquely distinguishes queries from each other within the search engine.

6. **Keynet structure.** See above for this structure.

6.3 Content Labels.

The structure of a content label is almost identical to that of a query. The only difference is that the first field contains a message type specifier appropriate to a content label rather than a query. This structure is used for registering a content label in the search engine repository.

6.4 Query Responses.

A query response structure is somewhat more complex than the structure for a content label because there will generally be several responses to a given query, and it is necessary to specify the query that is being answered using a given query response packet.

1. **Message type specifier.** This specifies that this structure contains a query response.
2. **Response count.** This is the number of responses that were generated by the query.
3. *Response* **rank.** This the sequence number of the response within the list of all responses to one query in order by their salience with the first response being the one judged to be most salient.
4. *Weight.* This is a measure of the salience of the response. Normally this will be expressed as a real number between 0 and 1. To store this number in an integer field, it is multiplied by a denominator (such as 1,000,000) that is specified in the ontology.
5. *Source identifier.* This is the source-generate query identifier from the original query.
6. **Engine identifier.** This is the engine-generated identifier from the original query.
7. **Keynet structure.** See above for this structure.

6.5 Protocol.

The protocol for queries is as follows. The query originator sends a query packet to the front-end of the search engine. The front-end immediately responds with an acknowledgement packet that includes the identifier assigned to the query by the search engine. The front-end

fills in missing field of the query packet and forwards the query to a randomly chosen home node for the query. The search engine uses the keynet algorithm to find and rank the desired responses. Each of these is sent back to the query originator from whichever search engine node finds it first. The query originator is responsible for collecting the responses, arranging them *in the correct order*, and *then displaying* them.

7 Summary.

KEYNET is a graph-oriented method for information object indexing and retrieval. Information Objects must be annotated with small semantic networks that represent their key concepts. A larger semantic network (part of a subject-specific ontology) determines which node and link types (basic concepts and relations) are considered pertinent to a subject during information object retrieval.

The query graph actually used for retrieval may substitute more general or specific concepts for those specified by the user. Retrieval does *not* match large components of the query graph against whole content labels. Instead, the query graph is *fragmented* into small probes of bounded size. These fragments are matched against content labels, and resulting retrieval sets of potentially relevant information objects are combined using fragment-oriented weights.

The graph representations, their fragmentation, and post-retrieval merging of information object sets associated with distinct fragments naturally introduce a "fuzziness" appropriate to information retrieval notions of relevance, and facilitate use of distributed or parallel processing resources (at appropriate stages).

The KEYNET system employs a number of optimizations to ensure that it scales up to large corpora and so that it has high performance. Fragmentation combined with pattern associativity of graph structures and linear hashing techniques produces tractable complexity of communications and computation, despite necessary isomorphism testing and index manipulation. The system is therefore compatible with the requirements for search engines needed in proposals for an NII.

References

[BF93] K. Baclawski and N. Fridman. M&M-Query: Database support for the annotation and retrieval of biological research articles. Technical Report NU-CCS-94-07, Northeastern University, College of Computer Science, 1993.

[BS94a] K. Baclawski and D. Simovici. An abstract model for semantically rich information retrieval. *Information Systems*, 1994. to be submitted.

[BS94b] K. Baclawski and J. E. Smith. A distributed approach to high-performance information retrieval. In *Proc. IEEE Sympos. Par. Distr. Processing*, 1994. Submitted.

[BS94c] K. Baclawski and J. E. Smith. High-performance, distributed information retrieval. Technical Report NU-CCS-94-05, Northeastern University, College of Computer Science, 1994.

[BS94d] K. Baclawski and J. E. Smith. A unified approach to high-performance, vector-based information retrieval. In *RIAO'94*, 1994. submitted.

[FBY92] William B. Frakes and Ricardo Baeza-Yates. *Information Retrieval / data structures and algorithms*. Prentice-Hall, Englewood Cliffs, NJ, 1992.

[FHL+91] Norbert Fuhr, Stephan Hartmann, Gerhard Lustig, Michael Schwantner, Konstadinos Tzeras, and Gerhard Knorz. AIR/X: A Rule-Based Multistage Indexing System for Large Subject Fields. In *Proc. User-Oriented Content-Based Text and Image Handling Conference (RIAO-91)*, pages 606–623, Centre de Hautes Etudes Internationales d'Informatique Documentaire, Paris, France, 1991.

[GIL] Government information locator service (GILS): Draft report to the Information Infrastructure Task Force. Available by anonymous ftp from 130.11.48.107 as /pub/gils.txt.

[EL93] Betsy L. Humphreys and Donald A.B. Lindberg. The UMLS project: making the conceptual connection between users and the information they need. *Bulletin of the Medical Library Association*, 81(2):170, Apr 1 1993.

[Jac93] Paul S. Jacobs. Using statistical methods to improve knowledge-based news categorization. *IEEE Expert*, pages 13–23, April 1993.

[Leh92] Fritz Lehmann. Semantic networks in artificial intelligence, parts I and II. *Computers and Mathematics with Applications*, 23(2-9), 1992.

[Lev91] Robert Levinson. A self-organizing pattern retrieval system and its applications. *International Journal of Intelligent Systems*, 6:717–738, 1991.

[Lev92] Robert Levinson. Pattern associativity and the retrieval of semantic networks. *Computers and Mathematics with Applications*, 23(6-9):573–600, 1992.

[LHM93] D.A.B. Lindberg, B.L. Humphreys, and A.T. McCray. The Unified Medical Language System. *Methods of information in medicine*, 32(4):281, Aug 1 1993.

[Sal89] Gerard Salton. *Automatic Text Processing*. Addison-Wesley, Reading, MA, 1989.

[SNA92] Ingeborg Sølvberg, Inge Nordbø, and Agnar Aamodt. Knowledge-based information retrieval. *Future Generation Computer Systems*, 7:379–390, 1991/1992.

An abstract model for semantically rich information retrieval

Kenneth Baclawski*
Northeastern University
College of Computer Science
Boston, Massachusetts 02115
(617) 373-4631
FAX: (617) 373-5121
kenb@ccs.neu.edu

Dan A. Simovici
University of Massachusetts
at Boston, Dept.of Mathematics
and Computer Science
Boston, Massachusetts 02125
(617) 287-6472
FAX (617) 265-7173
dsim@cs.umb.edu

March 31, 1994

Abstract

An abstract model for information retrieval is developed. The model is designed for semantically complex retrieval from a corpus of information objects in a single subject area. For example, the research papers or a corpus of images or data files relevant to a scientific discipline would be appropriate for this model. a knowledge model of ontology must be defined for the subject area, and objects must be labeled with data structures conforming to the ontology. The model supports high-performance information retrieval based on the vector space model for a corpus of several million information objects. The model also supports the construction of dictionaries or glossaries. A glossary can be used for query modifications in which a term of the query is replaced by its explanation in the glossary.

1 Introduction

With the expansion of the Internet and the development of new "Information Superhighways," computer-based communication is becoming the defining technology of this decade.

*This material is based upon work supported by the National Science Foundation under Grant No. IRI-9117030.

The amount of information that will be available over these new networks is immense: on the order of billions of objects and hundreds of terabytes of data. Information retrieval (IR) in such an environment is a monumental task but essential to its success. We propose an IR model, called KEYNET, that unifies and extends many commonly used IR mechanisms. A distributed architecture and indexing algorithm has been developed for high-performance IR using the KEYNET model[BS94b, BS94a]. The prototype system has achieved a throughput of 500 queries per second with a response time of less than a second for more than 85% of the queries[BS94c].

The KEYNET system is designed for IR from a corpus of information objects in a single subject area. It is especially well suited for non-textual information objects, such as scientific data files, satellite images and videotapes; although some kinds of textual document, such as research papers in a single discipline, can also be supported. With current technology, KEYNET can support very high-performance IR from a corpus having up to several million information objects at approximately the same level of performance as smaller corpora.

We begin by presenting the architecture of the KEYNET system. This will explain where the various kinds of information are located, the pathways for communication, and how a user interacts with a KEYNET search engine. The purpose of this section is to provide motivation for the definitions and algorithms that are developed in the later sections. Specifically, in sections 3 and 4, the concepts of a keynet ontology and a keynet conforming to an ontology are developed. Two important special cases of keynets are glosses or definitions and fragments. These are developed in sections 5 and 6, respectively. We end with a discussion of related

work.

2 The KEYNET Architecture

The purpose of KEYNET is to assist in retrieving information objects from a corpus of them. These information objects need not be textual and may be physically located anywhere in the network. Retrieval is accomplished by means of a *content label* for each information object. These content labels are stored in a repository at the KEYNET site. The structure of the content labels is specified by an information model or *ontology*. The content labels are indexed by means of a distributed hash table stored in the main memories of a collection of processors at the KEYNET site. These processors form the *search engine*. Each content label contains information about locating and acquiring the actual information object.

To see more precisely where all of these components reside, and how they are connected to one another, refer to Figure 1. The user's computer is in the upper left. A copy of the ontology is kept locally at the user site. As this will require at least several hundred megabytes of memory, it would generally be stored on a CD-ROM. The ontology is also the basis for the user interface to the search engine[BP93]. Queries must conform to the format specified by the ontology, and are sent over the network to a front-end processor at the KEYNET site. Responses are sent back over the network to the user's site, where they are presented to the user using the ontology.

At the KEYNET site, the front-end computer is responsible for relaying query requests to one of the search engine computers. The reason for having a front-end computer is mainly for distributing the workload but it also helps to simplify the protocol for making queries. The

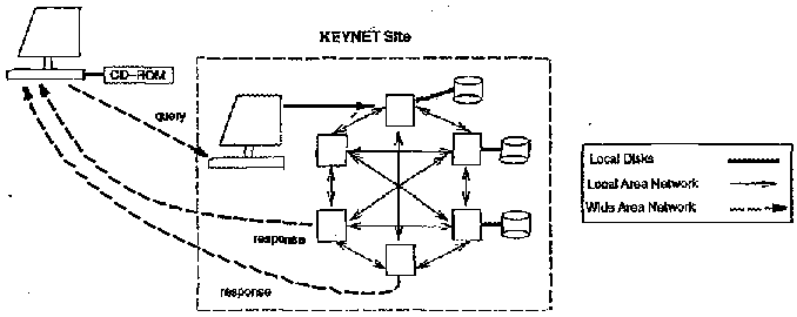


Figure 1: Architecture of KEYNET Search Engine

search engine itself is a collection of processors (or more precisely server processes) joined by a high-speed local area network. The search engine processors are called *nodes*. The repository of content labels is distributed on disks attached to some of the nodes. The index to the content labels is distributed among the main memories of the nodes.

The basic indexing strategy of KEYNET is to match probes (fragments of queries) with index terms (fragments of content labels). We now give an outline of the distributed algorithm that accomplishes this matching. For more details see [BS94a]. This algorithm can be characterized as a "scatter-gather" technique. Queries are sent to a front-end processor that forwards the query to a randomly chosen node of the search engine. This is the first scattering step. The node that is assigned the query is called the *home node* of the query.

At the home node, the query is broken apart into probe fragments using Algorithm 12 below. Each probe is then hashed using a standard algorithm. The hash value is in two parts. One part is a node number and the other part is the local hash value used at that

node. The local hash value and the query identifier are then sent to the node that was selected by the hash value. The result of hashing is to scatter the probes uniformly to all of the nodes of the search engine.

Upon receiving the local hash value of a probe, the node looks it up in its local hash table. An index term in the hash table that matches a probe is called a "hit." The hits are sent back to the home node of the query. This is the "gather" step of the algorithm. The home node then computes the similarity measure (currently the cosine measure) of each object in the collection, and the objects are ordered by the degree of similarity. The object identifiers of the most relevant objects are then sent back to the user.

The insertion of a new content label in the index is done in a manner very similar to the query algorithm. The same fragmentation, hashing and scattering algorithms are used for content labels as for queries. The only difference is that instead of matching entries in the hash table, index terms are inserted into the table.

3 Ontologies

Both queries and content labels are represented using a data structure called a *keynet* that is semantically intermediate between a keyword and a semantic network. The KEYNET model is derived from an earlier data model called the mu model[BS92]. Both the mu and KEYNET models are complex object models. In this section and section 4, keynets are formally defined, and examples are given to relate the formalism to commonly used mechanisms for IR. The underlying mathematical structure on which keynets are based is the *directed graph*[CLR90, Section 5.1], which we define as follows:

Definition 1 *A* directed graph *G is a quadruple $(V, E, \mathrm{src}, \mathrm{tar})$, consisting of sets V and E, called the* vertex *set and the* edge *set, respectively, and functions* src*: $E \to V$ and* tar*: $E \to V$, called the* source *and* target *functions, respectively.*

The elements of V and E (as well as of the lexicon L to be defined below) will generally have additional structure such as textual labels and informal explanations. The edge set E, in particular, will be labeled with *type* information. When drawing a directed graph, one uses boxes or dots for vertices and arrows for edges. Both the boxes or dots and the arrows are labeled with some of the additional structure belonging to the corresponding vertices and edges. For a directed graph G, we write $G.V$ for its vertex set, $G.\mathrm{src}$ for its source function, etc. The *dimension* of a directed graph is the number of edges it has. A *loop* is an edge e such that $\mathrm{src}(e) = \mathrm{tar}(e)$. The *in-degree* of a vertex v is the number of edges e such that $\mathrm{tar}(e) = v$. Similarly, one defines the *out-degree*. The *degree* is the sum of the in-degree and out-degree. Note that a loop is counted twice in the degree of a vertex.

Definition 2 *A* (directed) path *in a directed graph G, is a sequence of edges $e_1, e_2, \ldots, e_n$, where $n \geq 1$, such that for every i, $0 < i < n$, one has $\mathrm{tar}(e_i) = \mathrm{src}(e_{i+1})$. The* transitive closure *of G is the graph G^+ for which:*

1. *$G^+.V = G.V$;*
2. *$G^+.E$ is the set of paths of G;*
3. *for a path $p = (e_1, e_2, \ldots, e_n)$, $\mathrm{src}(p) = \mathrm{src}(e_1)$ and $\mathrm{tar}(p) = \mathrm{tar}(e_n)$.*

The reflexive closure *of G is the graph G^0 which is the same as G but with an additional loop at every vertex (representing a hypothetical path with no edges at the vertex). The* reflexive transitive closure *of G is the graph $(G^+)^0$. The* undirected graph *of G is the graph G^2 which has the same vertices as G, but it has twice as many edges: each edge of G has two copies in G^2, one is an exact copy, while the other has the source and target reversed.*

A directed graph G is *strongly connected* if G^+ is complete, i.e., every pair of (not necessarily distinct) vertices is linked by edges in both directions. A directed graph G is *connected* if G^2 is strongly connected and it is *acyclic* if G^+ has no loops.

The framework or schema for a KEYNET system is its *ontology*. The word ontology means "a branch of metaphysics relating to the nature and relations of being." Our use of the word is much more restrictive, dealing only with the nature of, and relationships among, concepts within a narrow subject area. Attempts to specify ontologies for scientific disciplines are very common, with most disciplines having some kind of subject classification scheme by this time. However, as Lakoff points out [Lak87] "human categorization is based on principles that extend far beyond those envisioned in the classical theory." As a result, simple classification methods leading to taxonomies of concepts are inadequate for expressing the rich variety of human categorization techniques.

The KEYNET system depends on having a background ontology that defines the structure and behavior of keynets. The following is the formal definition:

Definition 3 *A* keynet ontology *is a quadruple (G, L, S, I), consisting of a directed graph G, called the* schema *or* semantic network, *a set L called the* lexicon, *a relation $S \subseteq L \times G.V$*

such that for every $l \in L$, there is some $v \in G.V$ such that $(l, v) \in S$., and a distinguished acyclic subgraph I of G, whose edges are called the ISA or subcategory *edges.*

The vertex set $G.V$ of an ontology can be regarded as the set of *conceptual categories.* It consists primarily of the subject classification categories for the subject covered by the corpus. However, an element of $G.V$ can also be a property or attribute of an information object, such as an author, its year of publication, etc. Still another interpretation of the elements of $G.V$ is as parts of speech (as in natural languages) or as types (as in programming languages). One typically expects that $G.V$ will have around 100 or so elements.

The Unified Medical Language System (UMLS) [HL93] of the National Library of Medicine is an example of a subject-specific ontology. Although the UMLS is more complex than a keynet ontology, it does have a core ontology that satisfies the definition of a keynet ontology.

The edge set $G.E$ consists of *links* that join conceptual categories in $G.V$. As mentioned earlier, the links are generally grouped into *link types.* The links in one link type share a common semantics. Perhaps the best known example of a link type is ISA, which is used to define the concept hierarchy for a subject classification. Because of their importance, we have chosen to distinguish the subgraph of G consisting of the ISA edges, although one could, in principle, extract this information from the labels. Other link types include "part of," "elaboration of," "cause of," etc.

The lexicon L of a keynet ontology can be regarded either as concepts or as lexical terms. These are the *keywords* of the subject area. The relation S specifies the conceptual category (or categories) of the keywords. One can regard this relation as being either *specialization* or

instantiation. We will use these terms interchangeably, but in a particular ontology only one will be appropriate. Each term is required to be a specialization of some semantic category, and can be a specialization of several. A term is generally a word or phrase, but could also be a person, a number (e.g., a year), or a range of names or numbers. For example, in UMLS one of the conceptual categories is *plant* and one of its specializations is *potato*.

One reason for splitting concepts into the schema and lexicon levels is pragmatic. While there are only about 100 elements of $G.V$ and even fewer link types, there will typically be on the order of 100,000 elements of L. By defining links only at the schema level, where there are fewer conceptual categories to deal with, one makes it easier to understand the ontology. However, the main motivation for the separation of objects into types and terms is to provide direct support for the notion of an attribute-value pair. This notion has proven to be very effective both for IR and database query systems.

Example 4 In this example we introduce an ontology that allows us to define properties of figures of two-dimensional geometry. Consider the lexicon L that consists of the following terms:

point	acute angle	diagonal
line	right angle	circle
segment	obtuse angle	ellipse
plane	triangle	parabola
angle	quadrilateral	hyperbola

The graph of the ontology is given in Figure 2.

□

As mentioned earlier, a directed graph is drawn using boxes and arrows, suitably labeled. For example, in the UMLS, part of the schema specifies that an acquired abnormality can

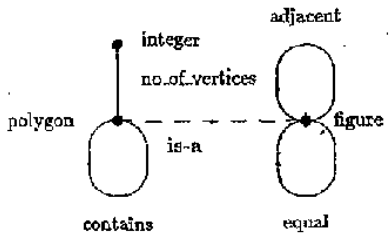
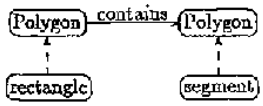


Figure 2: Graph of geometric ontology

be part of a plant:

Specializations of categories will be drawn with dashed lines. So one expresses that potato is a specialization of plant as follows:

As another example, using the geometric ontology of Figure2, one can express the fact that a rectangle contains a segment:



One commonly used IR mechanism is the use of attributes to identify documents. For example, the names of authors or words from the title. These are represented in an ontology with "Author" and "Title" concept categories. The lexicon would have the names of authors and (key)words from titles (or ranges of these if the number of authors or title words was too large). Ranges can also be used to express "wild card" matches.

4 Keynets

Intuitively, a keynet *conforms* to an ontology if it consists of copies of vertices, edges and terms of the ontology that related to one another the same way that the corresponding objects do in the ontology. Unfortunately, the notion of "the same way" requires some elaboration because of the possibility of conceptual categories inheriting properties via ISA relationships. More precisely,

Definition 5 *A* keynet K conforming *to an ontology* (G, L, S, I) *is a quintuple* (H, T, i, f, t), *where*

1. H *is a directed graph;*
2. T *is a set;*
3. $i: T \rightarrow H.V$ *is a one-to-one function, i.e., if* $t_1 \neq t_2$, *then* $i(t_1) \neq i(t_2)$;
4. $f: H \rightarrow G^0$ *is a pair of functions* $f.V: H.V \rightarrow G^0.V$ *and* $f.E: H.E \rightarrow G^0.E$ *such that for every edge* $e \in H.V$, *there are paths in* I^* *from* $f.V(H.\mathrm{src}(e))$ *to* $G^0.\mathrm{src}(f.E(e))$ *and from* $f.V(H.\mathrm{tar}(e))$ *to* $G^0.\mathrm{tar}(f.E(e))$;
5. $t: T \rightarrow L$ *is a function such that* $f.V \circ i \subset S \circ t$ *as relations, i.e., as subsets of* $T \times G^0.V$.

Although the definition above looks formidable, it can be expressed in a more intuitively clear way as follows: a keynet consists of a collection of conceptual categories from the ontology (allowing repetitions), some of which are instantiated, and some pairs of which are

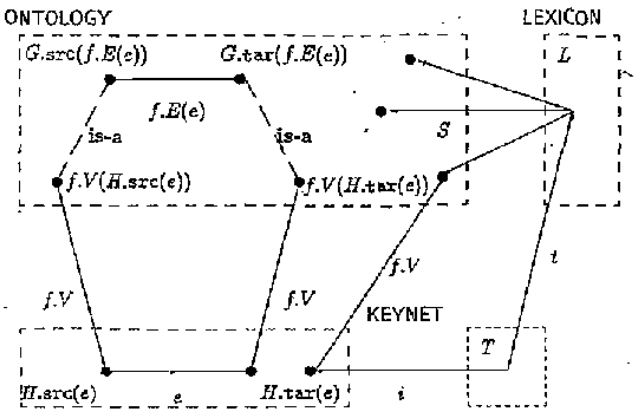


Figure 3: Keynet conforming an ontology

joined by links. The constraints in the definition express "type constraints" imposed by the ontology (see Figure 3).

For example, a bear could not instantiate the plant category, and a plant cannot be a part of an acquired abnormality. In addition, each occurrence of a conceptual category in a keynet can be instantiated at most once. To obtain multiple instantiations, one should use several occurrences of the conceptual category. For example, if an information object has several authors, the keynet would use one copy of the author category for each author.

In Figure 4, there is an example of a keynet conforming to the UMLS ontology.

5 Glossaries

A keynet ontology can be thought of as a language for expressing concepts in a specific subject area. A keynet is a statement conforming to this language. In the KEYNET system,

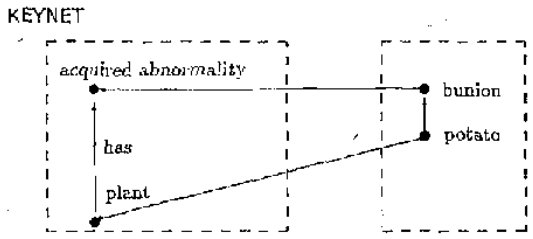


Figure 4: Can a potato get bunions?

we use keynets both for content labels of information objects and for queries to a corpus of information objects. Keynets are also used for two other notions that are useful for information retrieval: *glosses* and *fragments*. A gloss is a statement of the meaning of a concept in terms of other concepts. A fragment is a piece of a keynet used for indexing purposes.

We first discuss the notion of a gloss or explanation of a term. A typical gloss begins by specifying a more general concept, called its *genus proximus*, and then by means of various properties (known as the *specific difference*) distinguishes the concept from other concepts within the more general concept. The specification of the genus proximus is an example of an ISA relationship. The various properties are examples of links with other concepts in the lexicon. Thus a gloss is a kind of keynet where one vertex plays a special role.

Unfortunately, it isn't quite that simple, as a glance at a dictionary will show. A term can only be defined within the context of a particular part of speech. In a similar way, a keynet gloss explains a term only within the context of one of its conceptual categories. More

precisely,

Definition 6 *Let $O = (C, L, S, I)$ be a keynet ontology, and let (l, c) be an element of S. A gloss of the pair (l, c) is a pair (l_0, D), consisting of a keynet $D = (H, T, i, f, t)$ conforming to O, and a distinguished element $l_0 \in T$ such that*

1. *$f.V(i(l_0)) = c$;*

2. *for every $v \in H.V$, there exists a path in H^* from $i(l_0)$ to v.*

The distinguished element l_0 is the genus proximus of the term l that is being glossed. The first condition ensures that both l and l_0 (or more precisely l and $t(l_0)$) belong to the same conceptual category c of the ontology. When the category c is clear from the context, it will be omitted. The second condition above states that not only must the keynet D of a gloss of l be connected, but also every term in D must be a property either of l_0 or of one of its properties (recursively). Since a gloss is explaining l, it is reasonable to expect that it will not contain parts that are irrelevant to this purpose. *A keynet glossary* is a keynet ontology together with at least one gloss for each pair in its specialization relation.

Example 7 The notion of square can be glossed as a rectangle that has two adjacent equal sides. Assuming that the notion of side of a polygon has been defined, this gloss translates into the keynet shown in Figure 5.

□

Glossaries are very useful for query modification techniques. For example, one can modify a query by replacing one of its terms by its definition. This works because of the property of

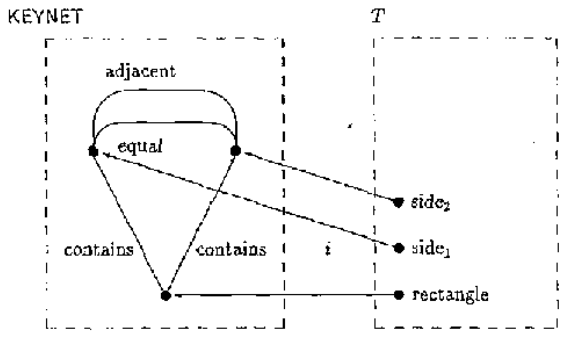


Figure 5: Keynet for the definition of square

a gloss known as *substitutability*: a term in any keynet can be replaced by one of its glosses.

Before we state this result, we need to define an *overriding union* operation on keynets.

Definition 8 *Let $O = (G, L, S, J)$ be an ontology, and let $K = (H, T, i, f, t)$ and $K' = (H', T', i', f', t')$ be two keynets conforming to O. Let A be a set, and suppose we have two one-to-one functions $j: A \to T$ and $j': A \to T'$ such that $f.V \circ i \circ j = f'.V \circ i' \circ j'$. Then there is a keynet $(H'', T'', i'', f'', t'')$ conforming to O and written $K \cup_A K'$ defined as follows:*

1. *$T'' = T \cup_A T'$, the union of T and T' with the images the elements of A identified. In other words, the disjoint union of $T - j(A)$, $T' - j'(A)$ and A.*

2. *H'' is the directed graph defined as follows:*

   (a) *$H''.V = H.V \cup_A H'.V$ via the one-to-one maps $i \circ j: A \to H.V$ and $i' \circ j': A \to H'.V$.*

   (b) *$H''.E$ is the disjoint union of $H.E$ and $H'.E$.*

(c) *$H''.\mathtt{src}$ is defined so that $H''.\mathtt{src}|H.E = H.\mathtt{src}$ and $H''.\mathtt{src}|H'.E = H'.\mathtt{src}$, and similarly for $H''.\mathtt{tar}$.*

3. *$t'': T'' \to L$ is defined so that*

$$t''(\ell) = \begin{cases} t(\ell) & \text{if } \ell \in T - j(A) \\ t'(\ell) & \text{if } \ell \in T' - j'(A) \\ t'(j'(\ell)) & \text{if } \ell \in A. \end{cases}$$

4. *$i'': T'' \to H''.V$ is defined by*

$$i''(\ell) = \begin{cases} i(\ell) & \text{if } \ell \in T - j(A) \\ i'(\ell) & \text{if } \ell \in T' - j'(A) \\ i'(j'(\ell)) & \text{if } \ell \in A. \end{cases}$$

5. *$f''.V: H''.V \to G.V$ is defined so that $f''.V|H.V = f.V$ and $f''.V|H'.V = f'.V$. This is well-defined because $f.V \circ i \circ j = f'.V \circ i' \circ j'$.*

6. *$f''.E: H''.E \to G.E$ is defined so that $f''.E|H.E = f.E$ and $f''.E|H'.E = f'.E$. This is well-defined because $H''.E$ is a disjoint union.*

We need to verify that $K \cup_A K'$ is a keynet conforming to O. Clearly, H'' is a well-defined directed graph, T'' is a well-defined set, and i'' is a one-to-one function. Let e be an element of $H''.E$. If $e \in H.E$, then there is a path in I^* from $f.V(H.\mathtt{src}(e))$ to $G''.\mathtt{src}(f.E(e))$ because K is a keynet conforming to O. Similarly for the condition on the tar functions. If $e \in H'.E$, then one uses the same argument with K replaced by K'. The fourth condition for a keynet is therefore satisfied. To check the fifth condition, let $l \in T''$. If $\ell \in (T - j(A)) \cup (T' - j'(A))$ the verification is immediate. If $\ell \in A$, then

$$f''.V(i''(\ell)) = f''.V(i'(j'(\ell))) = f'.V(i'(j'(\ell)) \in S(t'(j'(\ell))) = S(t''(\ell)).$$

This gives the last condition, and it follows that $K \cup_A K'$ is a keynet conforming to O.

Note that the overriding union operation above is not commutative in general. We use the overriding union operation to introduce the substitution of a gloss in a keynet as follows:

Theorem 9 *Let $O = (G, L, S, I)$ be a keynet ontology, let (l, c) be an element of S, and suppose that (l_0, D) is a gloss of (l, c), where $D = (H, T, i, f, t)$. If $K = (H', T', i', f', t')$ is a keynet conforming to O and $m \in T'$ satisfies $t'(m) = l$ and $f'.V(i'(m)) = c$, then $K \cup_{\{m\}} D$ is a well-defined keynet conforming to O, where the maps j and j' are defined by $j(n) = l_0$ and $j'(m) = m$.*

Proof. We need to verify the requirement of Definition 8. $f.V(i(j(n))) = f.V(i(l_0)) = c$, by definition of j and a gloss. $f'.V(i'(j'(m))) = f'.V(i'(m)) = c$ by definition of j' and the hypothesis. It follows that $f.V \circ i \circ j = f'.V \circ i' \circ j'$, and therefore $K \cup_{\{m\}} D$ is a well-defined keynet conforming to O, and the result follows.

Note that no use is made in the proof of the condition $t'(m) = l$. This condition ensures that the gloss is being substituted for the term that the gloss explains. However, any the gloss could be substituted for any term in the same conceptual category to produce a valid keynet even though this would not be meaningful. In other words, a substitution may be syntactically correct, but semantically meaningless.

6 Fragmentation

The next notion we consider is the basis for the indexing of information objects using keynets. To avoid a "combinatorial explosion" of possibilities for indexing, it is necessary to restrict

the attention to a special kind of keynet, defined as follows:

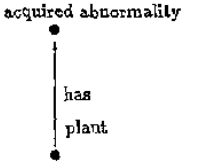
Definition 10 *A* fragment F *of a keynet* $K = (H, T, i, f, t)$ *is a pair* (A, B) *such that*

1. A *is a nonempty connected subgraph of* H;

2. $B \subseteq T$;

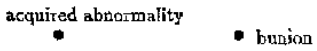
3. *The restrictions of* i, f *and* t *define a keynet structure* (A, B, i, f, t).

For example, this is a fragment of the keynet in Figure 4:



and here is another:

However, the keynet



is not a fragment because it is not connected.

A fragment having exactly one edge is called a *clasp*, while a fragment having exactly two edges is called a *double clasp*. We will later see in Theorem 13 that if one considers only

small fragments (i.e., no more than dimension 2), then the number of fragments of a keynet is at most linear in the size of the keynet. Thus there are very good pragmatic reasons for limiting ones attention to such fragments.

We now describe an algorithm that produces a list of all fragments for a given keynet K such that the fragments have dimension at most 2. The algorithm is more subtle than one would expect because of the possibility of loops and multiple edges in the keynet. We introduce the algorithm in two steps. We first give a fragmentation algorithm for directed graphs, and then use this algorithm to fragment keynets. The directed graph algorithm requires three kinds of iteration: over all vertices of the graph, over all outgoing edges emerging from one vertex, and over all incoming edges toward one vertex. The algorithm also requires a total ordering on all edges of the graph. For example, one might order the edges by their appearance in the representation for the graph.

Algorithm 11 *Given a directed graph G, to output every nonempty connected subgraph F of G such that F has at most two edges:*

Iterate over all vertices v of G:

1. *Output the subgraph consisting of just the vertex v.*

2. *Iterate over outgoing edges e from v, i.e., all e such that $\mathrm{src}(e) = v$:*

   (a) *Output subgraph consisting of the edge e and its two vertices.*

   (b) *Iterate over the outgoing edges f from v: If $(e < f)$ then output the subgraph consisting of the two edges e and f and their vertices.*

(c) Iterate over the incoming edges f toward v: If $(e \neq f)$ and $(\mathtt{src}(f) \neq v)$ and $((e < f)$ or $(\mathtt{tar}(e) \neq \mathtt{src}(f)))$ then output the subgraph consisting of the two edges e and f and their vertices.

3. Iterate over incoming edges e toward v and for each such edge, iterate over incoming edges f toward v: If $(\mathtt{src}(e) \neq v)$ and $(e < f)$ and $(\mathtt{src}(f) \neq v)$ and $(\mathtt{src}(e) \neq \mathtt{src}(f))$ then output the subgraph consisting of the two edges e and f and their vertices.

Proof. We proceed by cases. A subgraph F can have 0, 1 or 2 edges, and can have 1, 2 or 3 vertices:

1. If $\dim(F) = 0$, then F has exactly one vertex and it will be output exactly once in step 1.

2. If $\dim(F) = 1$, then F has exactly one edge. Let e be the edge, and let $v = \mathtt{src}(e)$. Then F will be output exactly once in step 2a.

3. The case $\dim(F) = 2$ breaks up into four disjoint subcases.

   (a) Suppose that F has one vertex. Then F is a double loop at the vertex. At most one of the two edges can be a loop in cases 2c and 3. If the loops are e and f, with e preceding f in the total order, then F will be output in step 2b.

   (b) Suppose that F has two vertices and one loop. Let v be the source and target of the loop. If the other edge is outgoing from v, then F will be output in step 2b because 2c excludes loops for the incoming edge. If the other edge is incoming

toward v, then F will be output in 2c because step 3 does not allow any loops in the subgraph.

(c) Suppose that F has two vertices and no loops. This is a "multiple edge" (or pair of parallel edges). If both edges have the same direction (i.e., the same source, and hence the same target), then F will be output in step 2c because step 3 does not allow the edges to have the same source. If the edges have opposite directions, then F will be output in step 2c. The condition in 2c prevents F from being output twice.

(d) Finally, suppose that F has three vertices. This is the typical case. Let v be the common vertex of the two edges. There are three subcases:

i. If the two edges are both outgoing from v, then the subgraph will be output in step 2b when the first edge precedes the second in the ordering on edges.

ii. If one of the edges is outgoing from v and the other is incoming, then the subgraph will satisfy the condition of step 2c when the first edge is outgoing and second is incoming.

iii. If the two edges are both incoming toward v, then as in the first subcase above the subgraph will be output exactly once in step 3.

The fragmentation algorithm above can easily be extended to deal with fragments of a keynet as follows:

Algorithm 12 *Given a keynet $K = (H, T, i, f, t)$, to output every fragment F of K such that F has dimension at most 2: Apply Algorithm 11 to H. For each subgraph A obtained*

in this way, iterate over every subset B of $i^{-1}(A.V)$. Each such subset defines a distinct fragment (A, B) of K.

It is easy to see that this algorithm produces every fragment of K. Because the map i is one-to-one, there are at most 3 elements in $i^{-1}(A.V)$, so there are at most 8 subsets B, hence at most 8 fragments of K for every subgraph A of H obtained from the directed graph algorithm.

The following results show that the number of fragments of a keynet does not grow too quickly as the size of the keynet increases.

Lemma 13 *Let G be a directed graph (connected directed graph) having v vertices and e edges. If the degree of any vertex of G is at most n, then the number of nonempty connected subgraphs of G having at most 2 edges is at most $v + e + e(n-1)$ (respectively, $1 + 2e + (e-1)n/2$).*

Proof. We proceed as in the proof of the fragmentation algorithm. There are clearly exactly v subgraphs having one vertex and no edges, and there are e subgraphs having one edge. Each connected subgraph having two edges has (at least) one vertex v such that v is incident on both edges of the subgraph. The number of such subgraphs having a given vertex v with this property is at most $\binom{deg(v)}{2}$. If there are no loops at v, then there will be exactly this many such subgraphs centered at v. If there are loops at v, then some subgraphs are counted more than once in the bound, so it will not be reached. If there is a multiple edge at v, then the subgraph has two vertices that can be used as center, so the bound will not be reached in the graph as a whole. In any case, since $deg(v) \leq n$, we obtain the upper

bound $v-e+\binom{n}{2}v$ on the number of nonempty connected subgraphs having at most 2 edges. However, if e is only slightly larger than v, then $deg(v)$ cannot take the maximum value too often, so it should be possible to improve this bound.

Since each edge is incident on two vertices, we know that $\sum_v deg(v) = 2e$. Note that this formula is correct even if some of the edges are loops, since such edges count twice in the definition of the degree of a vertex. The number of connected subgraphs having 2 edges is bounded by the sum $\sum_v \binom{deg(v)}{2}$. Since we also know that $0 \leq deg(v) \leq n$ (respectively, $1 \leq deg(v) \leq n$), we can obtain an upper bound on the number of such subgraphs by maximizing the sum $\sum_v \binom{deg(v)}{2}$, subject to the linear constraints on $deg(v)$. For the rest of the proof we assume that G is connected. The general case is similar. It is easy to see that the maximum is achieved when as many vertices as possible have degree n, possibly one vertex has degree between 1 and n, and the rest of the vertices have degree 1. More precisely, if r is $e-1$ modulo $n-1$, then the maximum is achieved when $\lfloor \frac{e-1}{n-1} \rfloor$ vertices have degree n, one vertex has degree $r+1$ and the rest have degree 1. The maximum value is $\lfloor \frac{e-1}{n-1} \rfloor \binom{n}{2} + \binom{r+1}{2} \leq (e-1)n/2$. Since the number of vertices in a connected graph is at most $e-1$, the result then follows.

Theorem 14 *Let $K = (H, T, i, f, t)$ be a keynet (connected keynet) such that H has v vertices and e edges. If the degree of a vertex of H is at most n, then there are at most $2v + 4e + 8e(n-1)$ (respectively, $2 + 6e + 4ne - 4n$) fragments having dimension at most 2.*

Proof. This follows easily from the lemma. Each subgraph of H results in at most 2, 4 or 8 fragments depending on whether the subgraph has 1, 2 or 3 vertices, respectively. The

result then follows.

The actual fragmentation algorithm used by KEYNET employs an additional restriction on the fragments. If a vertex is instantiated in the keynet K but not instantiated in a fragment F, then we say that the vertex is a *variant* vertex in the fragment. In the KEYNET system, at most one variant is allowed in each fragment. This reduces the bound in Theorem 14 to $2v + 3e + 4e(n-1)$.

7 Related Work

While semantic networks from AI were influential in the development of the KEYNET model, its indexing technique is fundamentally based on the vector space model for IR [Sal89]. From the point of view of IR, KEYNET can be regarded as a mechanism for unifying a number of *different* IR mechanisms, and the KEYNET *system* can be regarded as a high-performance, distributed search engine that can be applied effectively to vector-based retrieval from a corpus of annotated documents. From this point of view, keynets serve the same role as subject classification categories, keywords and properties such as author, title or date of publication.

Despite their reputation in IR circles as cumbersome, inefficient and suitable only for small databases, at least one IR researcher has used knowledge-based indices successfully [FHL+91]. Fuhr *et al*'s AIR/X performs automatic indexing of documents using terms (descriptors) from a restricted vocabulary. Probabilistic classification determines indexing weights for each descriptor using rule-based inference. KEYNET differs from AIR/X in using a form of semantic network as part of the retrieval algorithm rather than in the extraction of suitable

terms to be used for indexing. The extraction of terms (or in our case clasps) from a corpus of textual information objects is an important problem. One of the projects related to KEYNET is an effort to automate the extraction of keynets from biological research papers, in particular the Materials & Methods sections of such papers. See [BFFP93, BFH+93a, BFH+93b]. However, such extraction is independent of the architecture and algorithm employed for retrieval (and isn't even possible for non-textual information objects).

After building a system similar to AIR/X, Jacobs [Jac93] determined that "the combination of statistical analysis and natural language based categorization is considerably better than either alone." His paper describes an automated set of statistical methods for pattern acquisition that operate inside a knowledge-based approach for news categorization (an area closely related to document classification and other IR tasks). Like AIR/X, Jacobs' system does not employ semantic networks in the IR engine. Another difference between KEYNET and Jacobs' system is that KEYNET is designed for a corpus of documents in a single subject area, where it is feasible to develop a subject-specific ontology. Developing an ontology for heterogeneous textual documents is a formidable task, many orders of magnitude larger than is feasible with current technology.

Some other examples of knowledge-based query modification systems include systems primarily for information retrieval such as those in [GS93, CD90, Har92, QF93] as well as systems designed for database systems such as the KNOWIT system of Sølvberg, Nordbø and Aamodt [SNA92] and the cooperative query answering system in [CC92]. All of these are front-end query modification systems added to an IR or database system. Such query

modification techniques can also be used with a KEYNET system; in fact, the concept of a glossary was developed specifically for this purpose. Furthermore, the high-performance of a KEYNET search engine is useful for supporting the much larger queries generated by query modification techniques.

8 Conclusion

The keynet model proposed in this paper is the successor to the nn model introduced earlier by the authors and it is designed for IR systems. The fundamental notions introduced in the paper are: ontology, keynet conforming to an ontology, gloss and fragment. The formalism unifies and generalizes many commonly used IR mechanisms. It supports semantically rich content labeling of information objects and retrieval mechanisms that are sensitive to the semantics of the objects. It is shown that a gloss can be substituted for the term it glosses resulting in another keynet in which the term is explained. An upper bound is computed for the number of fragments that can be extracted from a given keynet. This bound is important in limiting the number of index terms in high-performance retrieval systems using the keynet model.

References

[BF93] K. Baclawski and N. Fridman. M&M-Query: Database support for the annotation and retrieval of biological research articles. Technical Report NU-CCS-94-07, Northeastern University, College of Computer Science, 1993.

[BFFP93] K. Baclawski, R. Futrelle, N. Fridman, and M. Pescitelli. Database techniques for biological materials & methods. In *First Intern. Conf. Intell. Sys. Molecular Biology*, pages 21–28, 1993.

[BFH+93a] K. Baclawski, R. Futrelle, C. Hafner, M. Pesaitelli, N. Fridman, B. Li, and C. Zou. Data/knowledge bases for biological papers and techniques. In *Proc. Sympos. Adv. Data Management for the Scientist and Engineer*, pages 23–28, 1993.

[BFH+93b] K. Baclawski, R. Futrelle, C. Hafner, M. Pesaitelli, N. Fridman, B. Li, and C. Zou. M&M-Query: Materials & Methods knowledge base and query system. Technical Report NU-CCS-93-06, Northeastern University, College of Computer Science, 1993.

[BS92] K. Baclawski and D. Simovici. An algebraic approach to databases with complex objects. *Information Systems*, 17:33–47, 1992.

[BS94a] K. Baclawski and J. E. Smith. A distributed approach to high-performance information retrieval. In *Proc. IEEE Sympos. Par. Distr. Processing*, 1994. Submitted.

[BS94b] K. Baclawski and J. E. Smith. High-performance, distributed information retrieval. Technical Report NU-CCS-94-05, Northeastern University, College of Computer Science, 1994.

[BS94c] K. Baclawski and J. E. Smith. A unified approach to high-performance, vector-based information retrieval. In *RIAO'94*, 1994. submitted.

[CC92] Wesley W. Chu and Qiming Chen. Neighborhood and associative query answering. *Journal of Intelligent Information Systems*, 1(3/4):355–386, December 1992.

[CD90] H. Chen and V. Dhar. Online query refinement on information retrieval systems: A process model of searcher/system interactions. In *Proc. SIGIR*, pages 115–133, 1990.

[CLR90] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.

[FHL+91] Norbert Fuhr, Stephan Hartmann, Gerhard Lustig, Michael Schwantner, Konstadinos Tzeras, and Gerhard Knorz. AIR/X: A Rule-Based Multistage Indexing System for Large Subject Fields. In *Proc. User-Oriented Content-Based Text and Image Handling Conference (RIAO-91)*, pages 606–623, Centre de Hautes Etudes Internationales d'Informatique Documentaire, Paris, France, 1991.

[GS93] S. Gauch and J. Smith. An expert system for automatic query reformulation. In *J. Amer. Soc. Info. Sys.*, volume 44, pages 124–136, 1993.

[Har92] D. Harman. Relevance feedback revisited. In *Proc. SIGIR*, pages 1–10, 1992.

[HL93] B. Humphreys and D. Lindberg. The UMLS project: making the conceptual connection between users and the information they need. *Bulletin of the Medical Library Association*, 81(2):170–177, 1993.

[Jac93] Paul S. Jacobs. Using statistical methods to improve knowledge-based news categorization. *IEEE Expert*, pages 13–23, April 1993.

[Lak87] George Lakoff. *Women, Fire and Dangerous Things*. The University of Chicago Press, Chicago, IL, 1987.

[QF93] Y. Qiu and H. Frei. Concept based query expansion. In *Proc. SIGIR*, pages 160–169, Pittsburgh, PA, 1993.

[Sal89] Gerard Salton. *Automatic Text Processing*. Addison-Wesley, Reading, MA, 1989.

[SNA92] Ingeborg Sølvberg, Inge Nordbø, and Agnar Aamodt. Knowledge-based information retrieval. *Future Generation Computer Systems*, 7:379–390, 1991/1992.

KEYNET: Fast Indexing for semantically rich information retrieval

by

Kenneth Baclawski and J. Elliott Smith

College of Computer Science
Technical Report NU-CCS-94-06

Northeastern University
Boston

KEYNET: Fast indexing for semantically rich information retrieval

Kenneth Baclawski*and J. Elliott Smith
Northeastern University
College of Computer Science
Boston, Massachusetts 02115
(617) 373-4631
FAX: (617) 373-5121
{kenb,csmith}@ccs.neu.edu

December 7, 1993

Abstract

We propose an architecture and index structure for semantically rich information retrieval from a subject-specific collection of documents or other information objects. The technique assumes that information objects have been labeled using small semantic networks called keynets. The index structure is compatible with parallel machine architectures and scales up well, allowing very large collections to be searched very quickly using semantically complex queries.

1 Introduction.

With the expansion of the Internet and the development of new "Information Highways," computer-based communication is becoming the defining technology of this decade. A number of proposals have been made to build a coherent structure over these new high-bandwidth networks to convert them into a National Information Infrastructure (NII). For example, the proposed I-95 Information Market [T+93] proposes an infrastructure that facilitates the free-market purchase, sale and exchange of services. The amount of information that will be available on the I-95 or any other NII is immense: on the order of billions of objects and hundreds of terabytes of data. Finding information in such an environment is a monumental task but essential to the success of the infrastructure. Any solution to this problem must

*This material is based upon work supported by the National Science Foundation under Grant No. IRI-9117030.

satisfy two important requirements: it must be scalable to handle the large number of information objects and it must be semantically rich enough to support effective information retrieval (IR).

We propose an architecture and indexing strategy for a search engine that would satisfy these requirements. The KEYNET system would support information retrieval for a collection of information objects in a single subject area, such as a collection of biological research articles, a set of court cases, files containing remote geophysical sensor data, or even collections of software programs and modules. KEYNET assumes that the information objects in its collection have been annotated using small semantic networks of key concepts (keynets) that are consistent with a subject-specific concept ontology.[1] Although there are certainly important distinctions between semantic networks, knowledge frames and objects in an object-oriented database, none of these distinctions are important for the purposes of KEYNET, so they will be used interchangeably within the context of this article.

A keynet is similar to an abstract or review of a document both in size and in being separately accessible from its corresponding document. If good tools are available, it could be generated by the author of the document with no more effort than is now taken to write the abstract or to select the keywords. It may eventually be possible to use natural language processing techniques to generate keynets, but this is only possible for textual information objects. A third possibility is to have professional annotators construct the keynets. This is less costly than one might expect. The biological research literature consists of some 600,000 articles per year. It would cost less than $30 million per year to annotate this literature with keynets, a tiny amount compared to the cost of generating this literature in the first place[BF93].

To give a simple example of a keynet consider an imaginary paper entitled, "POOQ: A parallel, object-oriented query system." Let's say that the paper uses some known dynamic programming algorithms to optimize queries for use on parallel machine architectures. A keyword-based approach would classify this paper by using phrases such as "parallel algorithms," "object-oriented databases," "dynamic programming," and "query optimization."

Keywords have the advantage that they may include topically relevant words and phrases that do *not* appear in the documents themselves. Furthermore, some information objects (like images, scientific data and software) have no text that can reasonably be used by traditional IR technology. Keynets enrich the possible semantics as compared with keywords based on simple subject classification schemes by adding relationships between keywords. In addition to being more expressive than sets of keywords, keynets would also generally be larger, though still much smaller than the entire information object.

To describe a keynet for the imaginary POOQ paper above, we must first describe a hypothetical ontology for Computer Science. Suppose that one of the classes in this ontology is concerned with the concept of translation or transformation. The keynet for the POOQ paper could begin with an instance of the transformation class which is linked via an attribute

[1] We are misusing the word "annotate" slightly here. Although a keynet may refer to portions of its document, its role is to classify or abstract rather than to explain the portions of the document to which it refers.

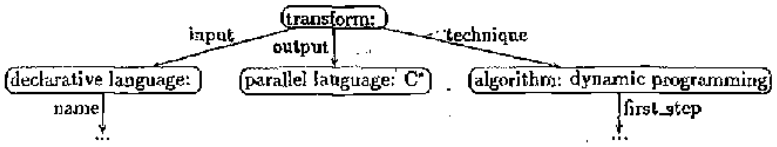


Figure 1: Example of part of a keynet

edge labeled "input" to an object having the subtype "declarative language," and so on. Using semantic networks the keynet would look something like that shown in Figure 1. In this figure, the label of each node consists of its type followed by its value (if any) separated by a colon. Since the output language is well-known, no further elaboration is needed. However, the input language, POOQ, is not well-known, so it must be specified further with attributes like its name and other attributes not shown.

The I-95 infrastructure proposal specifically mentions the need for "content labels [on information objects] to permit users to learn about available resources." These content labels are designed to deal with the problem of scaling up to a full-scale NII. Keynets would provide a solution to the design of such content labels. In spite of supporting the rich semantic content possible with semantic networks, the KEYNET system uses efficient indexing techniques based on hashing to make it a fast and scalable search engine.

Since the purpose of the KEYNET system is information retrieval, the paper begins with some background on IR in section 2. Despite the perceived inefficiency of semantic network techniques in IR, there are systems that use such methods, and we discuss such related work in section 3. We then describe the overall system architecture of KEYNET.

The KEYNET system is an information retrieval system for a subject area as specified in a database called the *ontology*. The ontology is discussed in section 5. The objects in the collection will be called *documents* even though, as noted earlier, the KEYNET system works equally well with information objects that are not documents in the usual sense of this term. The most common use of KEYNET begins with a user query. The query may be expressed using natural language text which is parsed to obtain a semantic network, or else a graphic interface may be used to express the query directly in terms of semantic networks. The same ontology is used for defining the structure of semantic networks of queries as that used for keynets. For example, the query "What systems use dynamic programming to generate C* code?" would be translated into a semantic network like that in Figure 2.

Having converted the original query into a semantic network, the next step is to break the network into small semantic networks having a bounded size. We call these fragments *probes*. Fragmentation is discussed in section 6. These fragments are indexed using a hash index structure defined in section 7. The keynets are also fragmented and the fragments inserted into the index as described in subsection 7.3. The search step consists of hashing the probes and looking for matches with fragments of keynets of documents. In the example,

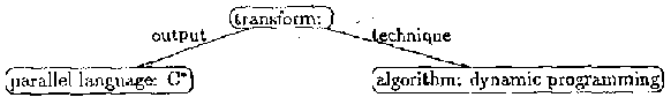


Figure 2: Semantic network of a query

the POOQ document would have fragments that match every probe of the query. The search algorithm is described in more detail in subsection 7.2.

We are in the process of developing a prototype KEYNET system for information retrieval on two document collections. The first is a small document collection and ontology prepared by the Biological Knowledge Laboratory [BFH+93] at Northeastern University. The second will be a full-size, randomly-generated document collection. The first collection will be used for testing *recall* and *precision*, while the second collection will be used for testing database performance. For a discussion of techniques for rapid generation of random databases see [GEBW93].

2 Information Retrieval.

Information retrieval systems are primarily concerned with the problem of providing mechanisms for a user to select a small set of relevant documents (or parts of documents like chapters, figures, tables, etc.) from a large document collection. This objective differs from database management in a number of ways. Generally speaking, IR systems are not concerned with answering detailed queries about the content of the documents in the collection. On the other hand, whereas a database system can answer detailed queries very precisely, most are not capable of answering the vaguely worded queries about relevance that an IR system can handle.

A database system takes for granted that a query is precisely stated, and the issue becomes how efficiently the query can be evaluated. By contrast, since IR queries are not required to be precise, one measures performance in a different manner. The two most common general measures of retrieval quality in IR research are called *recall* and *precision*. The former is the ratio of documents retrieved versus the number of available documents relevant to the query, i.e., the fraction returned out of all desirable documents (a measure of alpha risk or type I error). The latter is the ratio of the number of relevant documents retrieved versus the total number of documents retrieved, or the useful fraction of what was actually retrieved (beta risk or type II error).

Recall and precision presume that categories assigned by human experts are correct, complete, and well-specified; yet emergent concepts are seldom clearly formulated. An ideal information retrieval mechanism must not only capture *poorly* expressed concepts, but also somehow adapt as ideas change; its conceptual ontology must evolve over time.

Most commercial IR systems are based on a boolean model of relevance. The query terms

are matched to keywords or to words in the document content, and relevance is determined by the satisfaction of a boolean expression specified by the user. A variety of techniques *such as word stemming, truncation,* thesauri and lexicons have been used to extend this model[Sal88].

In contrast to boolean methods, the so-called "vector" methods use a notion of relevance that is less sharply defined: a document has a degree of relevance (salience) rather than simply being relevant or irrelevant. Documents are represented as points in a multidimensional *vector space. One can then compare* documents to each other as well as to queries. One commonly used measure is the cosine of the angle between the points regarded as vectors [Sal89]. Other possible measures of salience include path distance between nodes in a graph, or the number of levels that must be traversed to connect two categories in a hierarchy of abstractions.

While vector methods use probability and statistical methods to improve retrieval effectiveness, they are the same as boolean methods in their reliance on simple linguistic units, such as combinations of words or phrases, as the basis for retrieval. Since fragments of natural language do not always communicate a concept unambiguously for every combination of speaker, listener, writer or reader, retrieval errors inevitably arise from irrelevant discourse. Consequently, to improve retrieval effectiveness IR systems often label documents using keywords or phrases that may never appear in the document itself.

Moreover, relevance requires relative judgement; material irrelevant for categorization may still be relevant for other user purposes. Retrieval that merely matches against text in a document body presumes concepts can be completely characterized by statistical correlations. Yet as Jacobs [Jac93] observes, "statistical methods must be an aid rather than a replacement for knowledge acquisition". Text statistics are best used to identify patterns that depend on specialized words and phrases not obvious to casual readers. Mechanisms that recognize complex ideas are better constructed by human experts, whose understanding of a concept may transcend language.

The power available in a contemporary pattern-matching IR system comes mostly from its lexicon. Efficiency motivates use of simple combinations of lexical categories, such as can be represented in regular expressions. Yet more complicated patterns and mechanisms provide a major advantage in category definition and retrieval despite the time and effort *required* to create them, rendering knowledge-based approaches preferable to statistical methods.

Unfortunately, knowledge-based approaches that utilize semantic networks are currently considered so inefficient that they are explicitly omitted from some IR textbooks. For example, [FBY92] dismisses them on the basis of "the amount of manual effort that would be needed to represent a large document collection."

3 Related Work.

Despite their reputation in IR circles as cumbersome, inefficient and suitable only for small databases, at least one IR researcher has used knowledge-based indices successfully [FHL+91]. Fuhr *et al*'s AIR/X performs automatic indexing of documents using terms (descriptors)

from a restricted vocabulary. Probabilistic classification determines indexing weights for each descriptor using rule-based inference.

After building a system similar to AIR/X, Jacobs [Jac93] determined that "the combination of statistical analysis and natural language based categorization is considerably better than either alone." His paper describes an automated set of statistical methods for pattern acquisition that operate inside a knowledge-based approach for news categorization (an area closely related to document classification and other information retrieval tasks).

Both of these systems attempt to automate the process of generating index terms. KEYNET, by contrast, is not directly concerned with how the index terms are obtained but rather in the data structures and indexing techniques that would make use of the index terms.

Several distinct families of databases for semantic networks have been developed. Such databases are often called knowledge-base systems. Some of the best known of these are: Conceptual Dependency, ECO, KL-ONE, NETL, Preference Semantics, PSN and SNePs (see [Leh92]). All of these support link types, frame systems and so on, but few if any explicitly concern themselves with performance measures familiar in traditional database work, such as minimizing the number of *disk accesses* required to retrieve complex structures (i.e., graphs assembled from multiple frames and their typed relations). Contemporary knowledge-base systems traverse semantic networks one frame/relation at a time. Unless the knowledge-base fits entirely in the main memory of a single processor, these traversals result in large amounts of virtual memory paging.

In [Lev91], Levinson describes a technique for pattern-oriented (graph) retrieval. Levinson's approach discovers common structures among graphs in a database, and then uses pattern associativity to reduce the required number of tests for subgraph isomorphism. A separate paper, [Lev92], compares techniques for subgraph isomorphism testing suitable for pattern-oriented retrieval. The baseline retrieval method described in [Lev92] considers a flat set of N networks with no pattern associativity information, in which each query requires N isomorphism tests. The first improvement indexes commonly occurring substructures in the graphs, and tests only a subset of the entire database. A second improvement creates a multilevel index of subgraph relationships between successive levels of substructures and the domain of graphs in the database (establishes a partial order). A final elaboration applies multilevel indexing to connectivity and label information required during subgraph isomorphism testing (via the refinement method), rather than to the graph substructures themselves. This produces a tree of "node descriptors" in order of increasing specificity, with actual index pointers at its leaves.

In KEYNET we adapt subgraph isomorphism techniques to determine the relevance of documents that were previously annotated with keynets. Our approach compiles (fragments of) search paths through a tree of node descriptors into a hash table, reducing lookup computation (although increasing database construction overhead).

The KNOWIT system of Sølvberg, Nordbø and Aamodt [SNA92] embeds a semantic network in a front-end query refinement system. Queries to the ESA-QUEST bibliographic database are expanded according to a semantic model that describes the meaning of a concept entirely in terms of its relations to other concepts in the model. The KNOWIT system is a front-end

to a traditional IR system, whereas KEYNET uses a different kind of search strategy.

Chu and Chen [CC92] explain that cooperative query answering (CQA) substitutes retrieval terms to and from some abstract object representation (such as a type abstraction hierarchy). Separate frameworks can be used for grouping "data" (collections of objects with the same type and many common properties) and "knowledge" (relationships between objects of different types in specific problem domains), for example data by (type) class and knowledge by subject. (The "pattern instances" that link data and knowledge are mildly reminiscent of semantic network nodes.)

One example of CQA is "Neighborhood query answering" (NQA) which first generalizes query terms and then re-specializes them into a collection of "nearby concepts" in a compound query. Neighborhood specifiers such as "*morning*", "*afternoon*" or "*cross-country flight*" substitute for specific times or airline names. Chu and Chen present a formal system of rewriting rules and nearness measures for query relaxation in NQA.

Another example of CQA is "Associative query answering" (AQA) which traces behaviour dependencies among "cooperating objects" under a given subject. Virtual "pattern instances" with restricted scope and behaviour provide many-to-many *linkages* between data objects and knowledge subjects. Knowledge hierarchies based on generalization, composition or abstraction are used to support deductive reasoning that obtains information relevant to query construction but not explicitly contained in a user's request, using a logic-based rule language and inference engine, flexible goals, object contexts, and so on.

In general, CQA is a front-end to a traditional database management system. Moreover, the user is required to specify the degree of relaxation explicitly; it is not done automatically.

The EDS TemplateFiller system [SMHC93] applies Message Understanding (MUC) text-filtering techniques to the generation of knowledge frames for one or a few specific subject areas from entire texts (computer product announcements). TemplateFiller fills in slots for frames that exist in a predefined schema of templates, ignoring subjects that are not in the schema.

There are many other MUC-style systems; in fact, there is an annual competition among them. Such a system could automate the construction of the keynets for a collection of specialized textual documents. In fact, we are building a system of this kind for biological research papers [BFH+93].

4 System Architecture.

The KEYNET system is an information retrieval system for a subject-specific collection of *documents*. The subject area is defined by a database called the *ontology*. The ontology includes a number of components such as its schema, sublanguage grammar and thesaurus. The schema specifies the structure of knowledge frames and handles the regular features of knowledge in the subdiscipline. Less regular features are specified in the thesaurus which can specify relationships between concepts in the schema as well as between individual terms in the sublanguage. The sublanguage grammar is used for parsing natural language queries as well as for generating natural language from frames. The ontology is discussed in more

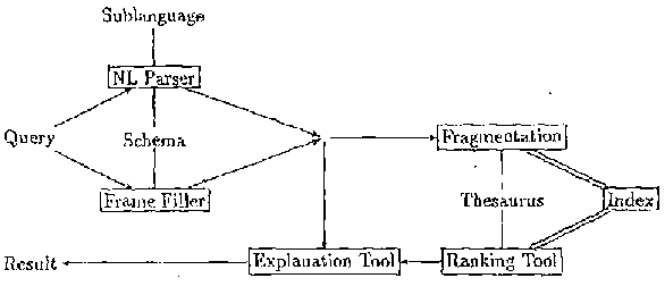


Figure 3: KEYNET System Architecture

detail in section 5.

The KEYNET system processes queries using a series of modules as in Figure 3. A user query may be formulated using natural language text which is parsed into frames using the Natural Language (NL) parser which, in turn, uses information in the Sublanguage database and the knowledge frame schema. The topic of NL parsing is beyond the scope of this article and is not described further. Alternatively, a user query may be directly expressed as knowledge frames using a graphical frame fill-in tool. Still another possibility (not shown in the diagram) is for the user to formulate the query using natural language and then to edit the resulting frames if they were not properly parsed by the NL parser.

However the knowledge frames are obtained, the next step is to break the frames into small semantic networks, called probes, each having a bounded size. This fragmentation is also done for the keynets of documents, and fragments thereby obtained will be called *index terms*.

The fragmentation of a query may also involve semantic "broadening" in which additional probes may be constructed by replacing concepts and terms by closely related concepts and terms. The thesaurus database specifies both the weight of such similarity links and the behavior of substitution during broadening. Broadening the query increases recall at the expense of precision. The user can specify how broadly or narrowly the query is to be interpreted by adjusting a threshhold value for semantic broadening.

The result of the fragmentation step is a large number of probes that can be indexed in parallel. This is shown in the diagram by using double arrows. The indexing step is analogous to the technique used by biologists to study the genome. A chromosome (a very long strand of DNA) is probed using small pieces of DNA which can attach to the chromosome only where they match in a precise fashion.

The KEYNET system separates the semantic broadening and indexing steps. As a result efficient hashing techniques may be employed rather than the somewhat less efficient

tree-structured indices. Tree-structured indices can be used to solve both the indexing and broadening steps using a single structure. For example, in a B-tree, entries that are alphabetically close are also close in the index. Techniques for indexing more than one dimension are known (for example, the hB-tree of Lomet-Salzberg [LS90]), but they are complex and are not yet commonly available in commercial systems. Semantic networks not only have a large number of "dimensions" but these dimensions are also not typically linear. Semantic proximity is very difficult to express or even to approximate using multidimensional vector spaces as in the vector models of IR. Accordingly, in the KEYNET system we use hash indices rather than tree indices. The cost of using hashing is that a much larger number of probes must be processed, but they can be processed in parallel.

The result of the indexing operation is a set of document identifiers each of which has a rough measure of relevance based on the number of probes that "hit" the document. This measure can be used to rank the documents. Alternatively, one can use more sophisticated graph isomorphism techniques. These more sophisticated techniques would use the thesaurus as well as the original query. As these techniques are beyond the scope of this article, they will be covered elsewhere.

The last step before presenting the result to the user is a tool for explaining how the documents were retrieved. This can be as simple as highlighting the passages that caused the retrieval. This technique works well only when no semantic broadening has occurred. More complex forms of matching will require not only highlighting but also natural language explanations.

Not shown in the diagram is the possibility of an additional form of user interaction known as *relevance feedback*. The user can indicate which documents in a set of retrieved documents are actually relevant to the original query. Such feedback can be used to modify the weights in the thesaurus, resulting in a customized thesaurus for each user.

Also not shown are the modules involved in constructing the ontology and the keynets of documents. These issues are the subject of the next section.

5 Ontology.

The word ontology literally means "a branch of metaphysics relating to the nature and relations of being." Our use of the word is much more restrictive, dealing only with the nature of, and relationships among, concepts within a narrow subject area. Attempts to specify ontologies for scientific disciplines are very common, with most disciplines having some kind of subject classification scheme by this time. However, as Lakoff points out,[Lak87], "human categorization is based on principles that extend far beyond those envisioned in the classical theory." As a result, simple classification methods leading to taxonomies of concepts are inadequate for expressing the rich variety of human categorization techniques.

The KEYNET system depends on having a background ontology that defines the structure and behavior of keynets. The Ontology Builder[BF93] is a system related to KEYNET that provides support for constructing and maintaining subject-specific ontologies that have much richer semantics than simple taxonomies.

One of the important features of the Ontology Builder is that it will automatically generate tools for entering keynets. We currently have a prototype called the M&M-Query system that has such a tool specialized for entering knowledge frames for the *Materials & Methods* sections of biological research papers[BF93]. This prototype furnished an important "proof of concept" for the feasibility of using such tools for annotating documents.

6 Fragmentation.

No matter how the original query is formulated, it is eventually converted into a graph (semantic network) which is then given to the Fragmentation Module. The output of this module is a set of small fragments called *probes*, a term borrowed from biology. This module is responsible for substitutions (broadening) and the computation of the probes.

Broadening is controlled by a weight associated with each possible substitution. Successive substitutions multiply the weights until a user-specified limit is reached. This prevents a combinatorial explosion.

After computing the substitutions, the resulting graphs are broken into probes. The simplest kind of probe is a single node. Each node is labeled with a type and possibly a value. Types are defined in the schema of the ontology, while values are defined in the lexicon. Roughly speaking, probes consisting of a single node correspond to the keywords of a traditional keyword-based IR system.

The next more complex probe is a pair of nodes connected by an attribute edge. Attribute edges are directed and have a label. The attribute labels are defined in the schema of the ontology. The most complex probe that is used is one having two attribute edges and two or three nodes. Note that fragments can consist of more than one node together with the edges (relationships) between them, so that any given node as well as any given relationship edge will generally occur many times in the set of all fragments.

The algorithm for fragmentation can be expressed as follows:

for every node n:
  output the node n
  for every variant n' of n:
    output the variant n'
  for every outgoing edge e:
    output the edge e with nodes n and t
    for every variant n' of the source node n:
      output the edge e with nodes n' and t
    for every variant t' of the target node t:
      output the edge e with nodes n and t'
  for every pair of (undirected) edges e and f incident on nodes t and u, respectively:
    output the triple (n, t, u)
    for every variant n' of the node n:
      output the triple (n', t, u)

for every variant t' of the node t:
    output the triple (n, t', u)
for every variant u' of the node u:
    output the triple (n, t, u')

Each probe has an associated weight. The weight is computed using weights taken from the ontology as well as substitution weights and possibly even weights specified in some way in the original query. The number of probes can be limited by specifying a lower limit below which probes will be discarded. Larger probes have a weight larger than the sum of the weights of its constituent parts since a match with such a probe would be much more meaningful than just matches with the individual nodes.

The fact that a match with a larger probe is more meaningful than one with a smaller probe suggests that indexing with larger probes is more useful than with smaller ones, in general. However, using larger probes expands the size of the index greatly with index terms that are much less likely to be matched. One must trade-off the inclusion of index terms that have high weight but little likelihood of being matched against index terms that have lower weight but greater likelihood of being matched.

Returning to the query example in Figure 2, the fragmentation step would produce six probes: one for each node, one for each edge, and one for the whole network. This assumes no broadening of the query. There are several ways one can broaden this query. The specific language, C*, could be broadened to any parallel language. The algorithm category could be replaced with a more general category such as "algorithm: search." Just using these two variants, the number of probes increases from six to twelve.

Even for a relatively small query, the fragmentation step can generate a large number of probes. However, for a fixed broadening limit, the number of probes is a constant times the size of the query. For a more precise statement of this see Theorem 1 and Corollary 2 in the Appendix.

7 Index Structure.

After fragmentation, the probes are hashed and matched with entries in the index. If a probe matches an entry in the index, then it is said to have *hit* the entry. Each entry consists of a document identifier and additional information to be discussed below. A document can be hit by many probes, and the sum of the weights of all probes that hit the document (suitably normalized) can be used to give an approximate measure of the relevance of the document to the query. Alternative measures of relevance are discussed in the next section.

In the rest of this section, we discuss how the index is structured and how operations are performed on the index. The details of the structure depend to some extent on the architecture of the machine to be used. The main distinction is whether the memory is globally shared (as in tightly coupled machines like the KSR-1) or local as in parallel computers like the MasPar or Connection Machine.

7.1 Data Structure.

The overall structure of the index is a hash table, with each component being called a *bucket*. The buckets, *in turn, have the structure of a cache*, a structure not often used in an index. For this reason, we call our index structure a *hash-cache index*.

Each probe is converted to a hash index using a standard hashing algorithm as in Knuth[Knu73, section 6.4]. For a probe p, let $h(p)$ be its hash value, a string of exactly n bits. We write $h_k(p)$ for the first k bits of $h(p)$, where $k \leq n$. Similarly, write $h^l(p)$ for the last l bits of $h(p)$. Write $w(p)$ for the weight of the probe p. Depending on the machine architecture, there will either be a fixed number of buckets in the table or the number of buckets can increase as needed. When the number of buckets can vary, we employ the dynamic hashing scheme due to Litwin[Lit80] which he has recently generalized to the case of *distributed systems*[LNS93]. When the number of buckets is fixed, the number is a power of two, say 2^m. The bucket for a probe p is then determined by $h_m(p)$. The rest of the hash value, $h^{n-m}(p)$, is then used for searching within the bucket. The same technique holds for the case of a variable number of buckets except that m is not necessarily the same for every bucket.

Within each bucket data is arranged as a tree, where $h_{n-m}(p)$ is the index. The tree must be balanced for a parallel machine architecture. Note that probes are not represented in the tree structure. One only has (parts of) hash values and document identifiers. For this to work, it is necessary for the number of bits n in the hash value to be large enough so that collisions will be very unlikely. Unlike traditional hashing algorithms, KEYNET can tolerate occasional collisions because of the redundancy inherent in using large numbers of probes. This optimization results in a significant reduction in the overall size of the index since (partial) hash values are much smaller than probes in general. It also improves the speed of indexing by replacing relatively costly string comparisons with fixed-size integer comparisons. For more detail about the probability model underlying this optimization, see the Appendix.

7.2 Searching.

To search for a keynet κ, one first performs substitutions on the nodes of κ, obtaining a list of variants for each node. One then fragments κ into a set of probes having at most three vertices, at most one of which is a variant node. See Corollary 2 in the Appendix for a bound on how many probes can be generated. Let $\mathcal{P}(\kappa)$ be the set of probes.

The next step is to compute $h(p)$ for every $p \in \mathcal{P}(\kappa)$. This can be done in parallel. The hash values are then sent to their buckets. More precisely, at a bucket with address a, one collects the weights $w(p)$ and partial hash values $h^{n-m}(p)$ of the probes that satisfy $h_m(p) = a$. Each partial hash value is then used for searching within its bucket. This step uses well-known algorithms.

The tree searching can be done in parallel. However, in this case, one must face the problem that not all buckets will have the same number of probes. To process all the probes in parallel will require a time proportional to the maximum number of probes occurring in a

bucket. This can be large, and it wastes resources because only a few (or even just one) of the parallel processors will be doing useful work when the last few probes are being processed. However, because of the statistical nature of the algorithm, it is acceptable to discard a small number of probes. When this is done, the number of parallel processing cycles needed can be reduced to reasonable values. For example, if there are 1,000 processors and 2,000 probes, and if one is willing to discard 2 probes on average, then it suffices to process at most 7 probes at each bucket. See the discussion and table in the Appendix for details.

The result of each search within a bucket is a set (possibly empty) of document identifiers. Each document identifier is associated with the weight $w(p)$ of the probe. Sets of document identifiers at different buckets are then merged, with the weights being accumulated for each document. The document identifiers having the highest accumulated weight are then passed to the next module for post-processing.

7.3 Insertion.

Insertion of keynets into the index begins with a fragmentation step similar to the one for searching except that no substitutions are done on the nodes. The number of index terms is then limited by the bound in Theorem 1. Assuming that the knowledge frames are not too large, one can expect that the number of index terms will be no more than about 4 or 5 times the number of nodes in the keynet. The index terms are then hashed as before, but now the partial hash values and document identifiers are inserted into the buckets.

There can, of course, be more than one document identifier associated with the same hash value. However, if the number of these exceeds a specified limit, then the document identifiers are dropped from the index, although the partial hash value is not. Instead, the partial hash value is associated with a marker to indicate that document identifiers were deleted.

A hash value with a large number of associated identifiers does not discriminate enough for it to be useful for retrieval. Traditional IR systems refer to nondiscriminating words as "stop words," and standard lists of stop words are available. Index terms that have too many associated identifiers will be called *stop terms*.

If documents are labeled by keynets having about 50 nodes, then there would be around 200 or so index terms per document. A collection of one million documents will then have over 200 million probes. However, when the nondiscriminating index terms are dropped, the number of terms will be substantially smaller, perhaps only 50-100 million. The resulting index should take up less than a gigabyte of memory. If each document requires 30-100K bytes, then the the index will be about two orders of magnitude smaller than the document collection itself.

When a bucket overflows, there are several strategies that can be used. If expandable hashing is being used, one simply allocates a new bucket (or buckets) and redistributes the trees into the new buckets. If there is a fixed number of buckets, then occurrences of identifiers will be selectively deleted from the bucket in a manner similar to a cache. The details of the caching algorithm are beyond the scope of this article. The effect of selective deletion of identifier occurrences is to "forget" index terms that are less useful. This does not

mean that the documents indexed by the forgotten terms are no longer accessible, since each document will have many terms associated with it. Eventually, of course, all the references to a given document could be forgotten at which point the document would cease to be accessible. However, this would only happen after a relatively long period of time. It would require more than just a loss of interest in this particular document. It would mean that there was no longer any interest in any feature of the document.

7.4 Deletion.

Explicit deletion of documents from the index can be done using a technique similar to the one used for insertion. The only peculiar feature of deletion is that stop terms can change back to being ordinary index terms if enough documents are deleted. However, it may not be necessary to delete documents from the index. As noted in the subsection above, one can allow a document collection to increase in size even with a fixed size index. In this case, documents do not get abruptly removed, but rather gradually get less and less accessible as interest in its index terms declines.

8 Summary.

KEYNET is a graph-oriented method for document indexing and retrieval. Documents must be annotated with small semantic networks that represent their key concepts. A larger semantic network (part of a subject-specific ontology) determines which node and link types (basic concepts and relations) are considered pertinent to a subject during document retrieval.

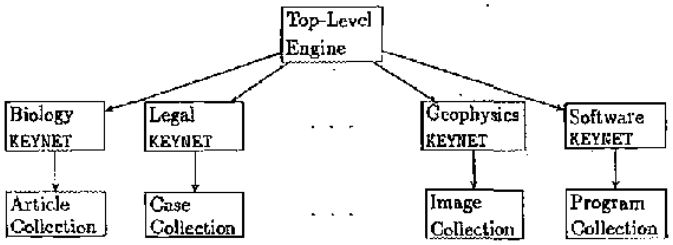
The query graph actually used for retrieval may substitute more general or specific concepts for those specified by the user. Retrieval does *not* match large components of the query graph against whole keynets of documents. Instead, the query graph is *fragmented* into small probes of bounded size. These fragments are matched against document keynets, and resulting retrieval sets of potentially relevant documents are combined using fragment-oriented weights.

The graph representations, their fragmentation, and post-retrieval merging of document sets associated with distinct fragments naturally introduce a "fuzziness" appropriate to information retrieval notions of relevance, and facilitate use of parallel processing resources (at appropriate stages). Although it was not the inspiration for KEYNET, it is interesting to compare it with human memory. Like human memory it is associative and semantically rich. It is also fault-tolerant: loss of a few probes during retrieval or loss of some of the buckets would make it somewhat harder to find a document but would not prevent it. It accomplishes this fault-tolerance by randomly distributing many index terms for each document throughout the entire index. Human memory is believed to have a similar feature. Human memory is highly parallel, and in the parallel version of the KEYNET system, the index has a fixed size with memories fading rather than abruptly disappearing.

The KEYNET system employs a number of optimizations to ensure that it scales up to large document collections and so that it has high performance. Fragmentation combined with

pattern associativity of graph structures and linear hashing techniques produces tractable complexity of communications and computation, despite necessary isomorphism testing and index manipulation. The system is therefore compatible with the requirements for search engines needed in proposals for an NII.

The proposed retrieval mechanism can be applied to instances of itself, producing a quasi-encyclopedic classification of documents. The following diagram suggests how one could organize search engines in the NII:



The top-level search engine appears to be a "hotspot" in this scheme, but it would not have to be consulted for every query. Most individuals would only use a few search engines and would not need to consult the top-level engine once the locations of these engines were known.

Using currently available technology, each search engine could support collections having *one million or more documents*. The top-level search engine could, in turn, support one million search engines. The entire structure would therefore index 10^{12} documents having an aggregate storage size of 10^{16} bytes, i.e., 10,000 Terabytes.

References

[BF93] K. Baclawski and N. Fridman. M&M-Query: Database support for the annotation and retrieval of biological research articles. In *Proc. ACM SIGMOD Conference*, 1993. Submitted.

[BFH+93] K. Baclawski, R. Futrelle, C. Hafner, M. Pescitelli, N. Fridman, B. Li, and C. Zou. Data/knowledge bases for biological papers and techniques. In *Proc. Sympos. Adv. Data Management for the Scientist and Engineer*, pages 23–28, 1993.

[CC92] Wesley W. Chu and Qiming Chen. Neighborhood and associative query answering. *Journal of Intelligent Information Systems*, 1(3/4):355–386, December 1992.

[FBY92] William B. Frakes and Ricardo Baeza-Yates. *Information Retrieval / data structures and algorithms*. Prentice-Hall, Englewood Cliffs, NJ, 1992.

[PHL+91] Norbert Fuhr, Stephan Hartmann, Gerhard Lustig, Michael Schwantner, Konstadinos Tzeras, and Gerhard Knorz. Air/x: A rule-based multistage indexing system for large subject fields. In *Proc. User-Oriented Content-Based Text and Image Handling Conference (RIAO-91)*, pages 606–623, Centre de Hautes Etudes Internationales d'Informatique Documentaire, Paris, France, 1991.

[GEBW93] J. Gray, S. Englert, K. Baclawski, and P. Weinberger. Quickly generating billion record synthetic databases. In *Proc. ACM SIGMOD Conference*, 1993. Submitted.

[Jac93] Paul S. Jacobs. Using statistical methods to improve knowledge-based news categorization. *IEEE Expert*, pages 13–23, April 1993.

[Knu73] Donald Knuth. *Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, Reading, MA, 1973.

[Lak87] George Lakoff. *Women, Fire and Dangerous Things*. The University of Chicago Press, Chicago, IL, 1987.

[Leh92] Fritz Lehmann. Semantic networks in artificial intelligence, parts i and ii. *Computers and Mathematics with Applications*, 23(2-9), 1992.

[Lev91] Robert Levinson. A self-organizing pattern retrieval system and its applications. *International Journal of Intelligent Systems*, 6:717–738, 1991.

[Lev92] Robert Levinson. Pattern associativity and the retrieval of semantic networks. *Computers and Mathematics with Applications*, 23(6-9):573–600, 1992.

[Lit80] Witold Litwin. Linear hashing: A new tool for file and table addressing. In *Proc. VLDB*, pages 212–223, 1980.

[LNS93] Witold Litwin, Marie-Anne Neimat, and Donovan A. Schneider. LH* Linear Hashing for Distributed Files. In *SIGMOD Record*, volume 22, pages 327–336, 1993.

[LS90] David B. Lomet and Betty Salberg. The hB-Tree: A Multiattribute Indexing Method with Good Guaranteed Performance. *ACM Transactions on Database Systems*, 15(4):625 658, December 1990.

[Sal89] Gerard Salton. *Automatic Text Processing*. Addison-Wesley, Reading, MA, 1989.

[SMHC93] H. Kelly Shuldberg, Melissa Macpherson, Pete Humphrey, and Jamil Corley. Distilling information from text: The eds templatefiller system. *Journal of the American Society for Information Science*, 44(9):493–507, 1993.

[SNA92] Ingeborg Sølvberg, Inge Nordbø, and Agnar Aamodt. Knowledge-based information retrieval. *Future Generation Computer Systems*, 7:379–390, 1991/1992.

[T+93] D. Tennenhouse et al. I-95 The Information Market. Technical Report MIT/LCS/TR-577, Massachusetts Institute of Technology, August 1993.

Appendix: Graph-Theoretic and Probabilistic Details

In this appendix, we sketch the derivation of some results from graph theory and probability that are useful for KEYKET.

Graph Theoretic Results

Theorem 1 *Let G be an acyclic, undirected graph with v vertices and e edges. If there are at most n edges incident on an vertex, then there are at most $1+2e+n(e-1)/2$ connected subgraphs having at most 3 vertices.*

To prove this, one first reduces to the connected case, in which case $e = v-1$. There are clearly $2e+1$ connected subgraphs having at most 2 vertices, so the problem is to bound the number of connected subgraphs having exactly 3 vertices. One can show that this number is *maximized* when all the vertices of G have either valence 1 or valence n (or as close to this as possible). There will be at most $(e-1)/(n-1)$ vertices with valence n, and each of these defines $\binom{n}{2}$ connected subgraphs having 3 vertices.

Allowing substitutions means that some of the vertices have *variants* that are distinguishable from the original vertex. Subgraphs are allowed to have at most one variant vertex. The following is an easy calculation given the one in the theorem above:

Corollary 2 *Let G be an acyclic, undirected graph with v vertices and e edges. If there are at most n edges incident on an vertex, and if each vertex has at most m variants, then there are at most $1+m+(3m+2)e+(3m+1)n(e-1)/2$ connected subgraphs having at most 3 vertices, where at most one of the vertices is a variant.*

Probability Models

If the hash function $h(p)$ is properly chosen, then the hash values will be approximately uniformly distributed over their range. If the hash value is n bits long, then this range is $[0, 2^n-1]$. If there are N items in the hash table, then the probability of randomly generating a hash value that corresponds to an item in the table is $N2^{-n}$. In particular, if a probe p does not match any item in the table, then it will accidentally match ("collide") with the hash value of an item in the table with probability $N2^{-n}$. This can be made arbitrarily small by choosing n large enough. For example, if N is 67 million (i.e., 2^{26}) and if n is 40, then the probability of a collision is $2^{-14} = 6 \times 10^{-5}$.

The probability model being used here is called the *finite occupancy process*. The remaining results are concerned with the distribution of a set of independent hash values distributed uniformly among a set of buckets. Let P be the number of hash values (i.e., the number of probes), and let B be the number of buckets. Suppose that one can only process up to N

probes per bucket. Let D_{PBN} be the number of probes that are not processed (and hence discarded), and let $FD_{PBN} = D_{PBN}/P$ be the fraction of all probes that are discarded. Both of these are random variables. We would like to compute the expectation (average) of these random variables:

Proposition 3 *Let P, B, N be positive integers with $B > 10$, and write α for the ratio P/B. Then the expectation of FD_{PBN} is approximately equal to*

$$\frac{1}{\alpha} \sum_{k=N+1}^{\infty} (k-N) \frac{\alpha^k}{N!} e^{-\alpha}.$$

The expectation is computed as follows. First approximate the finite occupancy process with the Poisson process. In Poisson process, the probability that there are k probes in a given bucket is $\frac{\alpha^k}{k!}e^{-\alpha}$. If $k \leq N$, then no probes are discarded; if $k = N+1$, then one probe is discarded; and so on. Thus the expected number of probes discarded in a single bucket is

$$\sum_{k=N+1}^{\infty} (k-N) \frac{\alpha^k}{k!} e^{-\alpha}.$$

The total number of probes discarded, D_{PBN}, is the sum of the number of probes discarded at each bucket. There are B buckets, and they are stochastically independent (unlike the case of the finite occupancy process). Hence

$$E(D_{PBN}) = B \sum_{k=N+1}^{\infty} (k-N) \frac{\alpha^k}{k!} e^{-\alpha}.$$

Finally, since $FD_{PBN} = D_{PBN}/P$, the result follows.

The following table gives values of N such that the expectation of FD_{PBN} is less than 0.01, 0.001 and 0.0001, for various values of the ratio P/B:

P/B	$E(FD) < 0.01$	$E(FD) < 0.001$	$E(FD) < 0.0001$
0.5	3	4	5
1	4	5	6
2	6	7	9
3	7	9	11
5	10	12	14
10	16	19	22
20	27	31	35

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application	:	Kenneth P. Baclawski
Serial No.	:	08/318,252
Filed	:	October 5, 1994
For	:	DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD
Attorney's Docket	:	NU-360XX

Group Art Unit: 2317

* *

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Honorable Commissioner of Patents and Trademarks, Washington, D.C. 20231 on 17 NOV 94.

By _____
Thomas A. Turano
Registration No. 35,722
Attorney for Applicant

* *

INFORMATION DISCLOSURE STATEMENT

Honorable Commissioner of Patents and Trademarks
Washington, D.C. 20231

Sir:

It is desired to cite for the record in this application the enclosed articles and U.S. patents listed on the attached copy of PTO Form #1449. The paragraph(s) marked below are applicable to this Information Disclosure Statement.

WENDEROTH, SCHURGIN, MACPHERSON & HEATH
TEL (617) 542-2290
FAX (617) 451-0313

Serial No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2317

[x] (1) The enclosed Information Disclosure Statement is being filed within three months of the filing date or within three months of the entry of the national stage of the above identified application. Accordingly, applicant(s) believe that no fee or certification is required.

[ ] (1a) Applicant(s) believe the enclosed Information Disclosure Statement is entitled to the benefit of 37 CFR §1.97(b)(3). Accordingly, applicant(s) believe that no fee or certification is required.

[ ] (1b) Pursuant to 37 CFR §1.97(c), the enclosed Information Disclosure Statement is being filed before the mailing date of a final action or a notice of allowance and is accompanied by:
[ ] a certification under 37 CFR §1.97(e); [ ] the fee set forth in §1.17(p).

PETITION UNDER 37 CFR §1.97(d)

[ ] (2) Pursuant to 37 CFR §1.97(d), applicant(s) hereby petition the Commissioner to consider the attached Information Disclosure Statement. Applicant(s) state that the issue fee has not been paid and that a certification under 37 CFR §1.97(e) is provided herein, along with the petition fee of $130.00 required under 37 CFR §1.17(i)(1).

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

CERTIFICATION UNDER 37 CFR §1.97(e)(1)

[ ] (3) The undersigned hereby certifies that each item of information contained in the attached Information Disclosure Statement was cited in a communication from a foreign patent office in a counterpart foreign application mailed ~~not~~ more than three months prior to the filing of this statement.

CERTIFICATION UNDER 37 CFR §1.97(e)(2)

[ ] (4) The undersigned hereby certifies that no item of information contained in the attached Information Disclosure Statement was cited in a communication from a foreign patent office in a counterpart foreign application or, to the knowledge of the undersigned, after making reasonable inquiry, was known to any individual having a duty of disclosure as set forth in 37 CFR §1.56(c) more than three months prior to the filing of this statement.

[ ] (5) The information disclosure fee of $200.00 required by 37 CFR §1.17(p) is believed to be due and is enclosed herewith.

The filing of this Information Disclosure Statement is not a representation by the undersigned as to personal knowledge of the contents of every word or phrase of the material enclosed or that reliance on other suitably trained professionals has not been made.

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

If a search report of a searching agency is enclosed identifying the nature of the relevance of each document, such a designation is deemed to satisfy Rule 98(a)(3) even if in a foreign language, since the few terms of relevance therein are deemed of universal cognizance. However, applicant does not necessarily adopt the position reflected by that report.

The Commissioner is hereby authorized to charge payment of any additional fees associated with this communication or credit any overpayment to Deposit Account No. 23-0804. Triplicate copies of this letter are enclosed.

Respectfully submitted,

KENNETH P. BACLAWSKI

By _____
Thomas A. Turano
Registration No. 35,722
Attorney for Applicant

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
Ten Post Office Square
Boston, Massachusetts 02109

Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: 17 Nov 94

TAT/dmk
56167.WP

Enclosures

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application	:	Kenneth P. Baclawski
Serial No.	:	08/318,252
Filed	:	October 5, 1994
For	:	DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD
Attorney's Docket	:	NU-360XX

Group Art Unit: 2317

* *

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Honorable Commissioner of Patents and Trademarks, Washington, D.C. 20231 on 17 Nov 99.

By _____
Thomas A. Turano
Registration No. 35,722
Attorney for Applicant

* *

INFORMATION DISCLOSURE STATEMENT

Honorable Commissioner of Patents and Trademarks
Washington, D.C. 20231

Sir:

It is desired to cite for the record in this application the enclosed articles and U.S. patents listed on the attached copy of PTO Form #1449. The paragraph(s) marked below are applicable to this Information Disclosure Statement.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

Serial No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2317

[X] (1) The enclosed Information Disclosure Statement is being filed within three months of the filing date or within three months of the entry of the national stage of the above identified application. Accordingly, applicant(s) believe that no fee or certification is required.

[ ] (1a) Applicant(s) believe the enclosed Information Disclosure Statement is entitled to the benefit of 37 CFR §1.97(b)(3). Accordingly, applicant(s) believe that no fee or certification is required.

[ ] (1b) Pursuant to 37 CFR §1.97(c), the enclosed Information Disclosure Statement is being filed before the mailing date of a final action or a notice of allowance and is accompanied by:
[ ] a certification under 37 CFR §1.97(e); [ ] the fee set forth in §1.17(p).

PETITION UNDER 37 CFR §1.97(d)

[ ] (2) Pursuant to 37 CFR §1.97(d), applicant(s) hereby petition the Commissioner to consider the attached Information Disclosure Statement. Applicant(s) state that the issue fee has not been paid and that a certification under 37 CFR §1.97(e) is provided herein, along with the petition fee of $130.00 required under 37 CFR §1.17(i)(1).

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

Serial No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2317

CERTIFICATION UNDER 37 CFR §1.97(e)(1)

[ ] (3) The undersigned hereby certifies that each item of information contained in the attached Information Disclosure Statement was cited in a communication from a foreign patent office in a counterpart foreign application ~~mailed not more than~~ three months prior to the filing ~~of~~ this statement.

CERTIFICATION UNDER 37 CFR §1.97(e)(2)

[ ] (4) The undersigned hereby certifies that no item of information contained in the attached Information Disclosure Statement was cited in a communication from a foreign patent office in a counterpart foreign application or, to the knowledge of the undersigned, after making reasonable inquiry, was known to any individual having a duty of disclosure as set forth in 37 CFR §1.56(c) more than three months prior to the filing of this statement.

[ ] (5) The information disclosure fee of $200.00 required by 37 CFR §1.17(p) is believed to be due and is enclosed herewith.

The filing of this Information Disclosure Statement is not a representation by the undersigned as to personal knowledge of the contents of every word or phrase of the material enclosed or that reliance on other suitably trained professionals has not been made.

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

Serial No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2317

If a search report of a searching agency is enclosed identifying the nature of the relevance of each document, such a designation is deemed to satisfy Rule 98(a)(3) even if in a foreign language, since the few terms of relevance therein are deemed of universal cognizance. However, applicant does not necessarily adopt the position reflected by that report.

The Commissioner is hereby authorized to charge payment of any additional fees associated with this communication or credit any overpayment to Deposit Account No. 23-0804. Triplicate copies of this letter are enclosed.

Respectfully submitted,

KENNETH P. BACLAWSKI

By _____
Thomas A. Turano
Registration No. 35,722
Attorney for Applicant

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
Ten Post Office Square
Boston, Massachusetts 02109

Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: 17 Nov 94

TAT/dmk
56167.WP

Enclosures

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

LB

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, DC 20231

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.
08/318,252	10/05/94	BACLAWSKI K.	NU960XX

EXAMINER
DAVIS, C

EQM1/0416
WEINGARTEN SCHURGIN GAGNEBIN & HAYES
TEN POST OFFICE SQUARE
BOSTON MA 02109

ART UNIT	PAPER NUMBER
2307	3

DATE MAILED: 04/16/96

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner of Patents and Trademarks

See attached office action.

PTO-90C (REV. 2/95) 1-File Copy

Office Action Summary

Application No.	Applicant(s)
08/315,252	Bielawski et al.
Examiner	**Group Art Unit**
Cheryl Lewis	3304

☒ Response to communication(s) filed on 10/06/94

☐ This action is FINAL.

☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11; 453 O.G. 213.

A shortened statutory period for response to this action is set to expire 3 month(s), or thirty days, whichever is longer, from the mailing date of this communication. Failure to respond within the period for response will cause the application to become abandoned. (35 U.S.C. § 133). Extensions of time may be obtained under the provisions of 37 CFR 1.136(a).

Disposition of Claims

☒ Claim(s) 1-14 is/are pending in the application.

Of the above, claim(s) _____ is/are withdrawn from consideration.

☐ Claim(s) _____ is/are allowed.

☒ Claim(s) 1-14 is/are rejected.

☐ Claim(s) _____ is/are objected to.

☐ Claims _____ are subject to restriction or election requirement.

Application Papers

☐ See the attached Notice of Draftsperson's Patent Drawing Review, PTO-948.

☐ The drawing(s) filed on _____ is/are objected to by the Examiner.

☐ The proposed drawing correction, filed on _____ is ☐ approved ☐ disapproved.

☐ The specification is objected to by the Examiner.

☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. § 119

☐ Acknowledgement is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d).

☐ All ☐ Some* ☐ None of the CERTIFIED copies of the priority documents have been

☐ received.

☐ received in Application No. (Series Code/Serial Number) _____.

☐ received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

*Certified copies not received: _____

☐ Acknowledgement is made of a claim for domestic priority under 35 U.S.C. § 119(e).

Attachment(s)

☒ Notice of References Cited, PTO-892

☒ Information Disclosure Statement(s), PTO-1449, Paper No(s). _____

☐ Interview Summary, PTO-413

☒ Notice of Draftsperson's Patent Drawing Review, PTO-948

☐ Notice of Informal Patent Application, PTO-152

SEE OFFICE ACTION ON THE FOLLOWING PAGES —

1. The following is a quotation of 35 U.S.C. 103 which forms the basis for all obviousness rejections set forth in this office action:

> A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.
>
> Subject matter developed by another person, which qualifies as prior art only under subsection (f) or (g) of section 102 of this title, shall not preclude patentability under this section where the subject matter and the claimed invention were, at the time the invention was made, owned by the same person or subject to an obligation of assignment to the same person.

2. Claims 1-17 are rejected under 35 U.S.C. 103 as being unpatentable over Chaturvedi, et al., "Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach", IEEE Transactions On Engineering Management, Vol. 41, No. 2, May 1994 and Houtsma et al., "Parallel Hierarchical Evaluation of Transitive Closure Queries", IEEE, 1991.

3. - With respect to claims 1, 6, 8, 12-13, and 17,

Chaturvedi discloses the distributed database system (Abstract, lin. 3) having the means which essentially comprise the same means as a home node (pg. 199, 2nd col., part B, example 4, node A). Chaturvedi discloses a plurality of query nodes (pg. 199, 2nd col., Part B, steps A.1-A.3, pg. 200, part C, Query Processing in TIF Environment) connected by a network (pg. 196, fig. 3). Chaturvedi discloses the fragmenting means (Abstract, lin. 1, pg. 195, Section II) and the means for a query from a user into a plurality of query fragments (pg. 196, fig. 2, 2nd col.,

section A, pg. 198, A. Illustrative Examples, Example 2, & No. 1-4). Chatrvedi discloses the local hash table means located on the query node (pg. 197, col. 1, par. 3, step 3, pg. 199, 2nd col. & Example 4). Chatrevdi discloses the object identifier means (pg. 198, A. Illustrative Examples, Example 2, lins. 1-4, & Site A-Site B). Chatrvedi discloses the means which essentially comprise the same means as the hashing means (Abstract, lins. 1-19).

Houtsma discloses the hashing means (2nd column, par. 3, & lin. 7) and the hashed query fragment means to have a first and second portion (pg. 133, adjacency, non-adjacency, Property 3.2, & Theorem 3.1). Houtsma discloses the hashed query fragment of the plurality of query fragments to a respective one of the query nodes indicating the first portion of each hashed query fragment (pg. 132, 2nd col., & lins. 1-9). Houtsma discloses the hashed query fragment to access data (pg. 130, 2nd col., par. 3, lins. 1-11).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the hashing means of Houtsma's teachings with the teachings of Chaturvedi because the hashing means could enable Chaturvedi's information retrieval means to provide the queried node with a query value and a query identifier during the query nodes hashing process.

4. - With respect to claims 2, 7, and 14,

Chaturvedi discloses the step of receiving at the claimed home node a query from the user (pg. 196, 1st col., par. 1, lins. 2-11, & figs. 2 & 3) prior to the step of fragmenting a query.

5. - With respect to claim 3,

Chaturvedi discloses the means to determine and return a measure of relevance and a predetermined degree of relevance between the accessed data and the query (pg. 199, Example 4, lins. 1-10, pg. 200, 1st col., base table T2, Node A, & Node B).

6. - With respect to claims 4 and 10,

Houtsma discloses the means which essentially comprise the same means as determining a measure or relevance by a cosine measure (pg. 130, 2nd col., par. 3, & lins. 5 & 6).

7. - With respect to claims 5 and 11,

Chaturvedi discloses where the hashed query fragment means comprises the bit means (pg. 199, 2nd col., & tuple 1).

8. - With respect to claim 15,

Chaturvedi discloses the three query level means (pg. 199, 1st col., steps 1, 2, and 4).

9. - With respect to claim 16,

Chaturvedi discloses where a query node returns a content label in response to a predetermined query level (pg. 198, 2nd col., Example 3).

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Cheryl Lewis whose telephone number is (703) 305-8750.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-9600.

PAUL V. KULIK
PRIMARY EXAMINER
GROUP 2300

Form PTO 948 (Rev. 10-93) U.S. DEPARTMENT OF COMMERCE - Patent and Trademark Office Application No. 318252

NOTICE OF DRAFTSPERSON'S PATENT DRAWING REVIEW

PTO Draftspersons review all originally filed drawings regardless of whether they are designated as formal or informal. Additionally, patent Examiners will review the drawings for compliance with the regulations. Direct telephone inquiries concerning this review to the Drawing Review Branch, 703-305-8404.

The drawings filed (insert date) 10/5/94 are:
A. ___ not objected to by the Draftsperson under 37 CFR 1.84 or 1.152.
B. X objected to by the Draftsperson under 37 CFR 1.84 or 1.152 as indicated below. The Examiner will require submission of new, corrected drawings when necessary. Corrected drawings must be submitted according to the instructions on the back of this Notice.

1. DRAWINGS. 37 CFR 1.84(a). Acceptable categories of drawings: Black ink. Color.
   ___ Not black solid lines. Fig(s) ___
   ___ Color drawings are not acceptable until petition is granted.

2. PHOTOGRAPHS. 37 CFR 1.84(b)
   ___ Photographs are not acceptable until petition is granted.

3. GRAPHIC FORMS. 37 CFR 1.84 (d)
   ___ Chemical or mathematical formula not labeled as separate figure. Fig(s) ___
   ___ Group of waveforms not presented as a single figure, using common vertical axis with time extending along horizontal axis. Fig(s) ___
   ___ Individual waveform not identified with a separate letter designation adjacent to the vertical axis. Fig(s) ___

4. TYPE OF PAPER. 37 CFR 1.84(e)
   X Paper not flexible, strong, white, smooth, nonshiny, and durable. Sheet(s) 3-12 COPY MACHINE MARKS OBJ.
   ___ Erasures, alterations, overwritings, interlineations, cracks, creases, and folds not allowed. Sheet(s) ___

5. SIZE OF PAPER. 37 CFR 1.84(f): Acceptable paper sizes:
   21.6 cm. by 35.6 cm. (8 1/2 by 14 inches)
   21.6 cm. by 33.1 cm. (8 1/2 by 13 inches)
   21.6 cm. by 27.9 cm. (8 1/2 by 11 inches)
   21.0 cm. by 29.7 cm. (DIN size A4)
   ___ All drawing sheets not the same size. Sheet(s) ___
   ___ Drawing sheet not an acceptable size. Sheet(s) ___

6. MARGINS. 37 CFR 1.84(g): Acceptable margins:

Paper size:			
21.6 cm. X 35.6 cm. (8 1/2 X 14 inches)	21.6 cm. X 33.1 cm. (8 1/2 X 13 inches)	21 cm. X 27.9 cm. (8 1/2 X 11 inches)	21 cm. X 29.7 cm. (DIN size A4)
T 5.1 cm. (2")	2.5 cm. (1")	2.5 cm. (1")	2.5cm.
L .64 cm. (1/4")	.64 cm. (1/4")	.64 cm. (1/4")	2.5 cm.
R .64 cm. (1/4")	.64 cm. (1/4")	.64 cm. (1/4")	1.5 cm.
B .64 cm. (1/4")	.64 cm. (1/4")	.64 cm. (1/4")	1.0 cm.

   Margins do not conform to chart above.
   Sheet(s) 3-12
   X Top (T) X Left (L) ___ Right (R) ___ Bottom (B)

7. VIEWS. 37 CFR 1.84(h)
   REMINDER: Specification may require revision to correspond to drawing changes.
   ___ All views not grouped together. Fig(s) ___
   ___ Views connected by projection lines. Fig(s) ___
   ___ Views contain *center lines*. Fig(s) ___

Partial views. 37 CFR 1.84(h)(2)
   ___ Separate sheets not linked edge to edge. Fig(s) ___
   ___ View and enlarged view not labeled separately. Fig(s) ___
   ___ Long view relationship between different parts not clear and unambiguous. 37 CFR 1.84(h)(2)(ii) Fig(s) ___

Sectional views. 37 CFR 1.84(h)(3)
   ___ Hatching not indicated for sectional portions of an object. Fig(s) ___
   ___ Hatching of regularly spaced oblique parallel lines not spaced sufficiently. Fig(s) ___
   ___ Hatching not at substantial angle to surrounding axes or principal lines. Fig(s) ___
   ___ Cross section not drawn same as view with parts in cross section with regularly spaced parallel oblique strokes. Fig(s) ___
   ___ Hatching of juxtaposed different elements not angled in a different way. *Fig(s)* ___

Alternate position. 37 CFR 1.84(h)(4)
   ___ A separate view required for a moved position. Fig(s) ___

Modified forms. 37 CFR 1.84(h)(5)
   ___ *Modified forms of construction must be shown in separate views.* Fig(s) ___

8. ARRANGEMENT OF VIEWS. 37 CFR 1.84(i)
   ___ View placed upon another view or within outline of another. Fig(s) ___
   ___ Words do not appear in a horizontal, left-to-right fashion when page is either upright or turned so that the top becomes the right side, except for graphs. Fig(s) ___

- REMOVE BOX AROUND FIG-FIG-1, 2)

9. SCALE. 37 CFR 1.84(k)
   ___ Scale not large enough to show mechanism without crowding when drawing is reduced in size to two-thirds in reproduction. Fig(s) ___
   ___ Indication such as "actual size" or "scale 1/2" not permitted. Fig(s) ___
   ___ Elements of same view not in proportion to each other. Fig(s) ___

10. CHARACTER OF LINES, NUMBERS, & LETTERS. 37 CFR 1.84(l)
   X Lines, numbers & letters not uniformly thick and well defined, clean, durable, and black (except for color drawings). Fig(s) 1-8

11. SHADING. 37 CFR 1.84(m)
   ___ *Shading used for other than shape of spherical, cylindrical, and conical elements of an object, or for flat parts.* Fig(s) ___
   ___ Solid black shading areas not permitted. Fig(s) ___

12. NUMBERS, LETTERS, & REFERENCE CHARACTERS. 37 CFR 1.84(p)
   X ___ Numbers and reference characters not plain and legible. 37 CFR 1.84(p)(1) Fig(s) 1-8
   ___ Numbers and reference characters used in conjunction with brackets, inverted commas, or enclosed within outlines. 37 CFR 1.84(p)(1) Fig(s) ___
   ___ Numbers and reference characters not oriented in same direction as the view. 37 CFR 1.84(p)(3) Fig(s) ___
   ___ English alphabet not used. 37 CFR 1.84(p)(2) Fig(s) ___
   X ___ Numbers, letters, and reference characters do not measure at least .32 cm. (1/8 inch) in height. 37 CFR(p)(3) Fig(s) 1-8

- DESCRIPTIVE MATTER OBJ (Fig. 1)

13. LEAD LINES. 37 CFR 1.84(q)
   ___ Lead lines cross each other. Fig(s) ___
   ___ Lead lines oblique. Fig(s) ___
   ___ Lead lines not as short as possible. Fig(s) ___

14. NUMBERING OF SHEETS OF DRAWINGS. 37 CFR 1.84(t)
   ___ Number appears in top margin. Fig(s) ___
   ___ Number not larger than reference characters. Fig(s) ___
   ___ Sheets not numbered consecutively, and in Arabic numerals, beginning with number 1. Sheet(s) ___

15. NUMBER OF VIEWS. 37 CFR 1.84(u)
   ___ Views not numbered consecutively, and in Arabic numerals, beginning with number 1. Fig(s) ___
   ___ View numbers not preceded by the abbreviation Fig. Fig(s) ___
   ___ Single view contains a view number and the abbreviation Fig.
   ___ Numbers not larger than reference characters. Fig(s) ___

16. CORRECTIONS. 37 CFR 1.84(w)
   ___ Corrections not durable and permanent. Fig(s) ___

17. DESIGN DRAWING. 37 CFR 1.152
   ___ Surface shading shown not appropriate. Fig(s) ___
   ___ Solid black shading not used for color contrast. Fig(s) ___

ATTACHMENT TO PAPER NO. 3 REVIEWER TGC DATE 11/1/94

PTO-948

Notice of References Cited

Application No.	Applicant(s)		
09/318,252	Baclawski		
Examiner		Group Art Unit	Page 1 of 1
Cheryl Lewis		2307	

U.S. PATENT DOCUMENTS

	DOCUMENT NO.	DATE	NAME	CLASS	SUBCLASS
A					
B					
C					
D					
E					
F					
G					
H					
I					
J					
K					
L					
M					

FOREIGN PATENT DOCUMENTS

	DOCUMENT NO.	DATE	COUNTRY	NAME	CLASS	SUBCLASS
N						
O						
P						
Q						
R						
S						
T						

NON-PATENT DOCUMENTS

	DOCUMENT (Including Author, Title, Source, and Pertinent Pages)	DATE
U	Chaturvedi, et al., "Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach", IEEE *Transactions On Engineering Management*, Vol. 41, No. 2	5/94
V	Houtsma et al., "Parallel Hierarchical Evaluation of Transitive Closure Queries", IEEE	4/1991
W		
X		

Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach

Alok R. Chaturvedi, Ashok K. Choubey, *Member, IEEE*, and Jinsheng Roan

Abstract— Different database fragmentation and allocation strategies have been proposed to partially replicate data in a partitioned, distributed database (DDB) environment. The replication strategies include database snapshots, materialized views, and quasi-copies. These strategies are 'static' and do not adapt to the changes in the data usage patterns. Furthermore, they often require expensive update synchronizations to maintain data consistency and do not exploit the knowledge embedded in the query history.

This paper describes a machine learning based time invariant fragmentation method (MLTIF) that acquires knowledge about the data usage patterns for each node. Based on this knowledge, MLTIF designs time invariant fragments (TIF) and schedules its allocation and selective update for a specified time period. Simulation is used to compare the effectiveness of the MLTIF approach with that of full replication, materialized views, and non replication strategies. Initial results indicate that for most normal operating conditions, the MLTIF approach can be effective.

Index Terms— Machine learning, distributed database, time invariant fragmentation, scheduling.

I. Introduction

In a distributed database system (DDBS), geographically dispersed databases are interconnected by a computer network, and their administration is done by a distributed database management system (DDBMS) in such a way that the distribution of logical and physical components of the databases are transparent to the user. The interest in (DDBS) research is motivated primarily by reliability, performance, and economic concerns. The reliability concern pertains to making the DDBS fault-tolerant; performance concerns include reducing query response time and increasing throughput; and economic concerns include reducing data communication and update synchronization costs.

To realize the above objectives, the database may be partitioned into a number of non-overlapping fragments and allocated over the network. Different strategies have been adopted to allocate data fragments in a DDBS. One data allocation strategy permits a single copy of the database to be stored in the network (no replication). Here, the database or its fragments are allocated to the nodes that minimize the overall system communication cost, query response time, and/or other criteria depending upon the objectives of the database designer [6], [18].

Another strategy involves storing multiple copies of all (complete replication) or a part (partial replication) of the database across the network [3], [5], [7], [10], [11], [17]. Although this reduces transmission costs and response time, it increases data redundancy, storage costs, and update costs. Several partial replication techniques have been proposed. Database snapshots are the read-only replica of a selected portion of the database [1]; materialized views are stored copies of the result of retrieving views from the database [4]; and quasi files are the portions of database stored in the cache memory at the user nodes [2].

Although these techniques have proven merits, they have the following shortcomings:

1) They are 'static' techniques, and do not adapt to the changes in the usage pattern in determining the content of the data. They assume the data usage is constant, while in reality, requirements tend to change frequently. Moreover, these techniques do not utilize the knowledge hidden in the query history of each node.
2) At a given time, data in a base table may be inconsistent with that of its copies. To overcome this problem, the above techniques require expensive update synchronizations [2], [4], [12], [14], [20].
3) In practice, most DDBS have attributes of an entity whose values do not change for lengthy periods of time. For instance, a typical customer relation of a commercial database has attributes, such as *customer number, name, address*, and *telephone*, whose values are relatively 'static.' This is an important property of data, and is called *time invariance*. This property is not exploited by any of the above methods.

The goal of this paper is to develop an adaptive method, based on the time invariance concept, that can autonomously detect data usage patterns from the query history of the given database, identify time-invariant fragments and their respective time windows, and allocate these fragments to the nodes such that data communication and update synchronizations costs are minimized. Re-stating the problem concisely,

Manuscript received November 25, 1992. Review of this manuscript was processed by Editor S. S. Brilgetic.

A. R. Chaturvedi is with the Krannert Graduate School of Management, Purdue University, West Lafayette, IN 47907.

A. K. Choubey is with the Information Systems—Data Management Applications, Bell Atlantic, Freehold, NJ 07728 USA.

J. Roan is with the Institute of Business Administration, National Chung Cheng University, Min-Hsiung Hsiang, Chia-Yi, Taiwan, R.O.C.

IEEE Log Number 9400869.

0018-9391/94$04.00 © 1994 IEEE

Given: (1) *A non-replicated distributed database, and*
(2) *Query history for each node in the network*
Find: (1) *Acceptable time intervals*
(2) *Time invariant fragments (TIF's)*
(3) *Schedule for TIF allocation and update.*

II. Design and Allocation of Time Invariant Fragments

A time invariant fragment (TIF) is a partition of a base relation whose contents are 'static' during a specified time interval. In other words, the values of each component attribute in a TIF are constant throughout this time interval. These attributes are called *Time-Invariant Attributes* (TIA's), and the remaining attributes, *Time-Sensitive Attributes* (TSA's). A formal definition of TIA and TSA is given below.

Let e_{ij} be the jth attribute in the ith tuple of a database relation R. Then the value sets of R and its jth attribute, a_j, can be defined, in terms of e_{ij}, as follows:

$R = \{e_{ij} \mid i = 1, 2, \ldots, m; j = 1, 2, \ldots, n\}$,
$a_j = \{e_{ij} | i = 1, 2, \ldots, m\}$.
a_j is a TSA for time interval T
if $\exists\, (v(e_{ij_t}) \neq v(e_{ij_{t'}}))$
$i \in \{1, 2, \ldots, m\}$
$t, t' \in \mathbf{T}$
$t \neq t'$
a_j is a TIA for time interval T
if $\forall (v(e_{ij_t}) = v(e_{ij_{t'}}))$
$i \in \{1, 2, \ldots, m\}$
$t, t' \in T$.
$t \neq t'$
where: $v(e_{ij_t})$ is the value of e_{ij} at time t,
m is the cardinality of R, and
n is the degree of R.

TIF's are constructed from the query history of the entire system. Some queries are periodic while others are *ad hoc*. The periodic queries, typically, follow patterns; for example, monthly sales statistics are inquired at the end of every month. Some periodic queries tend to appear together such as the group of queries required for the generation of a monthly sales report by geographic area, customer type, product type, and sales personnel. Some of these queries may access a large amount of common data. If the update of the database is not intensive in the time interval within which data access takes place, it could be beneficial to retrieve the common/shared time-invariant, remote data in one attempt.

The need to acquire knowledge about the retrieval patterns from query history suggests the use of a machine learning technique. One of the roles of machine learning is to seek to acquire knowledge from available data and use it to create new theories about the domain in question, in an entirely automated manner. Machine learning techniques employ a small number of extremely general induction strategies coupled with some basic domain knowledge. The domain knowledge may involve structural descriptions, procedural explanations, or even discoveries of new domain concepts. Due to the inductive nature of reasoning involved there is always some possibility of error. Consequently, most techniques allow for self-improvement through disconfirmatory feedback. Many learning strategies have been devised to solve problems in different domains [8]. Learning strategies include learning by being told, learning from example, learning from observations, case-based learning, analogical learning, and explanation-based learning.

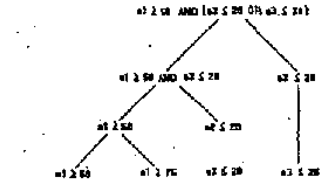


Fig. 1. Conceptual aggregation.

One of the induction based strategies, learning from observation, is adopted for the design of time invariant data fragments in a distributed database environment. The *technique*, called machine learning based time invariant fragmentation (MLTIF), has its roots in goal directed conceptual aggregation (GDCA), developed by Chaturvedi et al [8]. The creation of a TIF is a problem solving exercise for MLTIF. From the timestamps on the queries an initial time slice is determined. Each query in the history is then decomposed into sub-expressions. Next, patterns of data retrieval and modification is generated from the sub-expressions. Similar patterns are aggregated to form the most general concepts. An example to demonstrate this process is given below.

Example 1: Here we show how *MLTIF creates aggregate* concepts from queries. Suppose there are two queries, $Q1$ and $Q2$ in the query history. These queries are first decomposed into sub-expressions such as $a1 \geq 50, a2 \leq 20$. Using aggregation operators (AND, OR, etc.), these expressions are converted into higher level concepts till a single concept covers all data requirements for a relation (Fig. 1). Finally, the highest level concept is used to create a query to the base table for creating the fragment.

The concepts generated by MLTIF are evaluated using a cost-based evaluation function. These steps are repeated till the *least cost time slice and most general concepts are determined*. Finally, TIF's for the time slice, and for each node, will be created from the base tables using the most general concepts. The contents of TIF's are illustrated in Fig. 2.

The creation of proprietary TIF's for each node in the network could lead to the sub-optimality and computational intractability problems. When a number of nodes have high degrees of commonality between their respective TIF's, the likelihood of a large amount of common data being transmitted to individual nodes increases, as do the update synchronization costs. In addition, as the size of the network increases, the TIF approach may become *computationally intractable*. In order to reduce the costs and computational complexity, network nodes

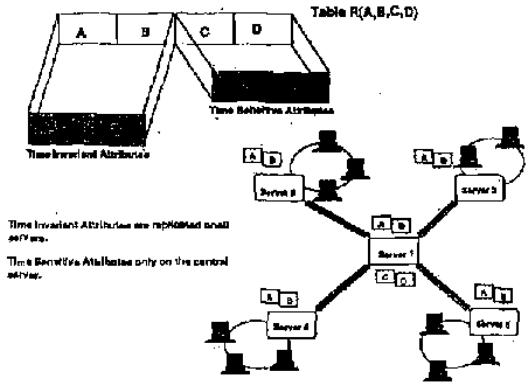


Fig. 2. The contents of TIF's.

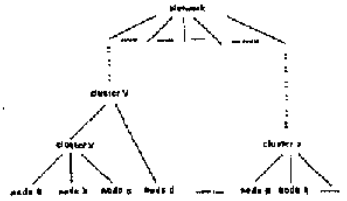


Fig. 3. TIF hierarchy.

are clustered according to the closeness of their locations and to the similarity in their query patterns (Fig. 3). Nodes with similar query patterns in a geographic area are assigned to the same cluster. For each cluster, one of the nodes may be selected or a new data server node may be created to store the common TIF. The common TIF may be retrieved by the member nodes in the cluster and may be refreshed when update synchronizations take place. Multiple levels of node clustering may be required. In such an event, two or more clusters may be combined into a larger cluster, and the TIF's are created for the member clusters in the same fashion as that for the nodes. The TIF hierarchy grows as long as it is computationally necessary. The increases in data retrieval costs and response times are offset by the decreases in TIF creation costs, data storage costs, and update synchronization costs.

III. Algorithm for Creation and Allocation of TIF's

The premise of the MLTIF approach is its capability of detecting the patterns that are likely to vary over time. As discussed above, a static methodology does not work in the ever-changing environment. A new approach that allows dynamic, conditions, and automatic detection of query patterns of a user node is required to provide the benefits discussed previously. Conceptual aggregation allows the computer program to aggregate observations (query statements) and provide meaningful explanation of aggregate concepts (query patterns) formed. Situations that are not currently handled are:

a) *Queries With Statistical Operations.* Attributes with statistical function such as SUM, MAX, and MIN, etc., are excluded from query statements. These operations usually involve simple values to be sent over from the remote base tables as a result of such functions. Our approach creates TIF's that contain fragments of remote base tables which would unnecessarily retrieve 'raw data' and increase communication cost. Therefore it would be more appropriate to set aside attributes with these statistical functions in the creation of TIF's.

b) *Updates With Insertion and Deletion.* Deletion and insertion of tuples are usually done periodically in batch mode. These update queries are excluded from this research without loss of generality.

c) *Data Movement.* Update with modification that causes data to move from one base table to another that is stored at different site. This is usually found in the change in the value of an attribute whose values are the basis of a horizontal partitioning of the database table. When the new value exceeds the value range of a base table, the entire tuple has to be removed from this table and added to an appropriate one. This type of update is an equivalence of a deletion followed immediately by an insertion.

The following algorithm is used by MLTIF to determine the TIF's for each site based on the query histories of the entire system.

Step 1. Classify the queries of each site into retrieval and update queries. Further classify the update queries into insertion, deletion, and modification.

Step 2. Classify the retrieval queries of each site into local-access and remote-access based on the location of data these queries address. Classify modification queries in a similar fashion.

Step 3. Transmit all the remote-access modification queries for each base table to their respective sites. For each base table, use these remote-access modification queries and the local-access modification queries to determine the time sensitive and invariant attributes.

Step 4. Remove time sensitive attributes from each remote-access retrieval query if these attributes are time sensitive in their remote base tables. All remaining attributes in this remote-access retrieval query are time invariant.

Step 5. Apply aggregation technique to construct the query for the creation of a TIF from a remote base table for each site. This is based on the entire remote-access retrieval queries for the base table at this site.

We now explain each step in detail.

Step 1. The type of database access of each query is identified. In SQL query language, a query start with a 'SELECT' is a retrieval, an 'UPDATE' a modification, an 'INSERT' an insertion, or a 'DELETE' a deletion query. Only retrieval and modification queries are used in the following steps for the determination of TIF's. Insertion and deletion queries are discarded.

Step 2. A retrieval or modification query in a partitioned, non-replicated database environment may refer to a data attribute of one of the following three types:

A. The attribute and all the data values are locally stored in a base table.

B. The attribute is not stored in any local base table.

C. The attribute is stored in a local base table but the referred data values are not stored or partially stored in this table. The data values that are not locally stored may spread over more than one remote base tables.

A type A attribute is either an attribute that is included in a vertically partitioned local base table or one that is used as the only attribute for horizontal partitioning of the original non-partitioned table and the values of this attribute in the local base table are the super-set of the required values of this attribute in the query statement. A type B attribute is an attribute that is not stored in the local base table which is a fragment of a vertically partitioned table. A type C attribute is an attribute whose values in the local base table are not the super-set of the required values in the query statement. This can be a result of a horizontal or a mixed partitioning of the original table.

All attributes in the SELECT part (selecting attributes) of a retrieval query or in the SET part (modifying attributes) of a modification query, or in WHERE part (restricting attributes) of both types of queries will be assigned one of these three types accordingly.A query is 'local-access' if all its attributes are of type A, is 'remote-access' if all its attributes are of type B, and otherwise is 'mixed-access'. Dealing with a mixed-access query is not as straightforward, as it is successively decomposed into multiple sub-queries till each subquery can be classified as a local-access or a remote-access query.

All attributes of a mixed-access query are first assigned into one of the two groups: group A in which all attributes are of type A and group B in which all attributes are of type B. An attribute of type C is assigned to both groups if the location of the base table containing its value cannot be identified from the database fragmentation conditions. An attribute that is not used to horizontally partition a database is usually of this type unless other restricting attributes in the same query statement can be used to assist the identification of the data source locations. We will exclude the selecting or modifying attributes with no restricting attribute stored in the same group. For a vertically partitioned database table, this situation implies that join operations are required for this query. For a horizontally partitioned database table, this means that there is not data to be accessed from this group of attributes. The union of all the local attributes in group A is still local and a query statement can be constructed to retrieve these data values from the local base table. This query is 'local-access'. A 'remote-access' query statement can be constructed from attributes in group B in the same fashion for the remote data of mixed-access query attributes.

Step 3. Remote-access modification queries are transmitted to the sites of the base tables they modify. The union of modifying attributes of both remote-and local-access queries for each base table will include some values that would vary in this time slice. These attributes are therefore 'time sensitive'. The supplement set of the attributes of this base table is 'time invariant'. The information of time sensitive attributes of a base table is sent back to all the sites with queries that modify this base table.

Step 4. Since a time sensitive restricting attribute will cause invalid selection of data fragment, this attribute should be eliminated from the remote-access retrieval queries. All time sensitive selecting attributes are dropped from these query statements. A remote access retrieval queries becomes invalid if there is no restricting or selecting attributes left in the query after the removal of time-sensitive attributes. Therefore, all the attributes in the queries are time-invariant, and will be used as the basis for constructing query statements for the creation of TIF's. The reason we did not further consider insertion and deletion queries after Step 1 is now clear. These two types of queries always change the content of base tables.

Step 5. At each site, all queries now consist of time-invariant attributes only. The remote access retrieval queries are grouped by base tables and for each base table all the queries are decomposed into sub-expressions. MLTIF takes all the sub-expressions corresponding to all the restricting attributes of the queries for a base table and aggregate them to form a higher level concept of the restricting attributes iteratively until no further aggregation is required. This final highest concept is used to construct the query to create a TIF of that remote base table. The restricting attributes that are not part of selecting attributes are added into the selecting attribute list because the restricting condition of a restricting attribute is an aggregation of the individual conditions. TIF's of the same domain and

structure are integrated into a single TIF. All the TIF's for a site can be constructed in this manner. An example illustrating *MLTIF* was presented in Section I.

A. Illustrative Examples

We now illustrate this procedure by two examples. The first example shows the creation of a TIF in a horizontally partitioned database environment and the second example shows the creation of a TIF in a vertically partitioned database environment.

Example 2: Suppose two sites in a network each stores a horizontally partitioned database table as follows:

CUST(CNUM, CNAME, CITY, CYTD_ORDER, CREDIT_LINE, CREDIT_USED)

Site A: stores the fragment with CITY = 'A', and
Site B: stores the fragment with CITY = 'B'.
The query histories are as follows:

Site A:

1. SELECT CNUM, CNAME, CYTD_ORDER
   FROM CUST
   WHERE CYTD_ORDER > 1000000
   AND CITY = 'A'
2. SELECT CNUM, CNAME, CYTD_ORDER
   FROM CUST
   WHERE CYTD_ORDER > 1500000
   AND CREDIT_LINE < 100000
3. SELECT CNUM, CNAME, CREDIT_LINE, CREDIT_USED FROM CUST
   WHERE CREDIT_LINE > 150000
4. UPDATE CUST
   SET CYTD_ORDER = CYTD_ORDER - 200
   WHERE CNAME = 'IBM'
   AND CITY = 'A'

Site B:

5. SELECT CNUM, CNAME, CREDIT_LINE
   FROM CUST
   WHERE CREDIT_LINE > 200000
6. UPDATE CUST
   SET CREDIT_USED = CREDIT_USED-200 WHERE CNUM = 'N321'
7. INSERT
   INTO CUST (CNUM, CNAME, CITY, CYTD_ORDER, CREDIT_LINE, CREDIT_USED)
   VALUES (N938, 'ABC INC.', 'B', 10000, 1000000, 0)

The creation procedure of the TIF's of CUST relation for both sites is as follows:

Step 1:
Classify query type.
Insert queries: 7
delete queries: none
modification queries: 4, 6
retrieval queries: 1, 2, 3, 5
query 7 will be discarded since it is an INSERT query.

Step 2: Classify local-/remote-access queries.

Retrieval queries:
Site A:
Local-access: 1, 2, 3 /* because it is transparent whether
Remote-access: 2, 3 CYTD_ORDER > 1500000 and CREDIT_LINE < 100000 in query 2 are local or remote

Site B:
Local-access: 5
Remote-access: 5

Modification queries:
Site A:
Local-access: 4
Remote-access: none /* because CITY = 'A' cannot be divided into two fragments */

Site B:
Local-access: 6
Remote-access: 6

Step 3: Identify time sensitivity.
Site A: Time-sensitive: CYTD_ORDER, CREDIT_USED
Time-invariant: all other attributes
Site B: Time-sensitive: CREDIT_USED
Time-invariant: all other attributes

Step 4: Remove time-sensitive attributes from remote-access retrieval queries.
Site A: remote-access query
Query 2: SELECT CNUM, CNAME, CYTD_ORDER
FROM CUST
WHERE CYTD_ORDER > 1500000
AND CREDIT_LINE < 100000
Query 3: SELECT CNUM, CNAME, CREDIT_LINE
FROM CUST
WHERE CREDIT_LINE > 150000
Site B: Remote-access retrieval query
Query 5: SELECT CNUM, CNAME, CREDIT_LINE
FROM CUST
WHERE CREDIT_LINE > 200000

Step 5: Construct the queries to create TIF's
MLTIF generates the following remote queries to create TIF's for both sites.
Site A: SELECT CNUM, CNAME, CREDIT_LINE
FROM CUST
WHERE CYTD_ORDER > 1500000
AND (CREDIT_LINE < 100000 OR CREDIT_LINE > 150000)
Site B: There is only one remote-access query, MLTIF will not generate a new query.

Example 3: Suppose two sites in a network each stores a vertically partitioned database table as follows:

CUST(CNUM, CNAME, CITY, CYTD_ORDER, CREDIT_LINE, CREDIT_USED)

Site A: stores the fragment **CUST(CNUM, CNAME, CYTD_ORDER)**
Site B: stores the fragment **CUST(*CNUM*, CITY, CREDIT_LINE, CREDIT_USED)**

The query histories are as follows:

Site A:

1. SELECT CNUM, CNAME, CYTD_ORDER
   FROM CUST
   WHERE CYTD_ORDER ≥ 1000000

2. SELECT CNUM, CYTD_ORDER, CREDIT_LINE
   FROM CUST
   WHERE CYTD_ORDER ≥ 1000000
   AND CREDIT_LINE < 100 000
   AND CITY = 'A'
3. SELECT CNUM, CNAME, CYTD_ORDER,
   CREDIT_LINE
   FROM CUST
   WHERE CYTD_ORDER ≥ 10000000
   AND CREDIT_LINE < 500000
   AND CITY = 'B'
   Site B:
4. SELECT CNUM, CNAME, CREDIT_LINE,
   CREDIT_USED, (CREDIT_USED / CREDIT_LINE)
   FROM CUST
   WHERE CREDIT_USED ≥ CREDIT_LINE * 0.9
5. UPDATE CUST
   SET CREDIT_USED = CREDIT_LINE - 2000
   WHERE CNUM = 10555
6. SELECT CNUM, CNAME, CYTD_ORDER,
   CREDIT_LINE, CREDIT_USED
   FROM CUST
   WHERE CNUM = 10555
7. INSERT
   INTO CUST(CNUM, CNAME, CITY,
   CREDIT_LINE, CREDIT_USED, CREDIT_ORDER)
VALUE (10056, 'ABC INC.', 'NEW YORK', 10000, 0, 0)

Step 1: Classify query type.

insert queries: 7
delete queries: none
modification queries: 6
retrieval queries: 1, 2, 3, 4, 5
query 7 will be discarded since it is an INSERT query.

Step 2: Classify local-/remote-access queries.

Retrieval queries:

Site A: Local-access: 1, 2, 3
Remote-access: 2, 3

Site B: Local-access: 6
Remote-access: 6

(Note: Query 4 is discarded because the remote set contains no restricting attribute.)

Modification queries:

Site A: None.

Site B: Local-access: 5
Remote-access: none

Step 3: Identify time sensitivity.

Site A: Time-sensitive: none
Time-invariant: all attributes

Site B: Time-sensitive: CREDIT_USED
Time-invariant: all other attributes

Step 4: Remove time-sensitive attributes from remote-access retrieval queries.

Site A: remote-access queries
Query 2: SELECT CNUM, CREDIT_LINE
FROM CUST
WHERE CREDIT_LINE < 100000 /*because CYTD_ORDER >='
AND CITY = '*A*' 1000000 is not in remote set*/

Query 3: SELECT CNUM, CREDIT_LINE
FROM CUST
WHERE CREDIT_LINE < 500000
AND CITY = '*B*'

Site B: remote-access query
Query 6: SELECT CNUM, CNAME, CYTD_ORDER
FROM CUST
WHERE CNUM = 10555

Step 5: Construct the queries to create TIF's.

MLTIF generates the following remote queries to create TIF's for both sites.

Site A: SELECT CNUM, CREDIT_LINE, CITY
FROM CUST
WHERE CREDIT_LINE < 500000
AND (CITY = '*A*' OR CITY = '*B*')

(Note: CITY is added into the selecting attribute list.)

Site B: There is only one remote-access query. MLTIF will not generate a new query.

B. TIF's for Retrieval Queries With Two-way Join

MLTIF processes two-way join queries in the following manner (all the other steps remain the same):

A. Determine TIF content by evaluating two-way join retrieval queries one by one.

Step A.1 Split the two-way join query into two single-table sub-queries (by dropping the join condition clause and splitting the selecting attributes) for the tables involved. Send each sub-query, along with the join condition clause to the nodes containing a copy of the base table involved. Some nodes might receive both sub-queries.

Step A.2 Evaluate sub-queries. Determine from the local base table the join attribute values. Put these values in a set, called *unique join-value set*. Transmit the set to the nodes receiving the other sub-query. Every node receiving a sub-query should have remote join values for the other sub-query.

Step A.3 Take the union of the remote unique join-value sets and the local unique join-value set, if any, at each node. Determine the data items satisfying the original two-way join query from the resultant set and the local base table.

Example 4: Suppose the two nodes in a network store data as follows:

$T1(\underline{a}, b, c, d, e, f)$ where a is a primary key,
$T2(\underline{f}, g, h)$ where f is a primary key,
Node A: stores the following base table fragment of T1 with $e = 'A'$,

	$\underline{a}$	b	c	d	e	f
tuple 1	*C001*	*Acme*	1000	200	*A*	S01
tuple 2	*C003*	Bree	500	100	*A*	S01
tuple 3	*C004*	Giant	100	100	*A*	S02

and base table T2:

	f	g	h
tuple 1	S01	Cecil	F
tuple 2	S02	Sam	M
tuple 3	S03	Georgia	M
tuple 4	S04	Diane	F

Node B: stores the following base table fragment of T1 with e ='B'.

	a	b	c	d	e	f
tuple 1	C002	Doris	2000	800	B	S03
tuple 2	C005	Gray	800	600	B	S04

The query history at each node is as follows:
Node A:

1. SELECT a, b, c
   FROM T1
   WHERE c > 1000
2. SELECT a, b, c, d, e
   FROM T1
   WHERE b = 'Gray'
3. SELECT a, b, c
   FROM T1
   WHERE e ='A'
4. UPDATE T1
   SET c = c − 100
   WHERE a = '001'

Node B:

1. SELECT a, b, c, d
   FROM T1
   WHERE c <= 500
2. SELECT $T1.a, T1.b, T1.c, T2.f, T2.g, T2.h$
   FROM T1, T2
   WHERE T1.c >= 1000
   AND $T1.f = T2.f$
   AND $T2.h$ = 'F'
3. UPDATE T1
   SET d = 500
   WHERE d < 500

MLTIF determines time-invariant fragments in the following manner:

There are no two-way join queries at node A while query 2 of node B is a two-way join retrieval query. The query is first decomposed into two single-table queries for base tables T1 and T2 as follows:

Sub-query B.2.1: SELECT a, b, c, f
FROM T1
WHERE T1.c >= 1000

and

Sub-query B.2.2: SELECT f, g, h
FROM T2
WHERE T2.h = 'F'

The join condition is $T1.f = T2.f$.

Note that the join attribute f is included in the selecting attribute list of both Sub-queries so that the original query can be recreated.

Since both nodes have a fragment of base table T1, Sub-query B.2.1 and the join condition are submitted to node A. Sub-query B.2.2 will also be submitted to node A with the join condition because base table T2 is stored only at node A.

At Node A: The only condition clause in Sub-query B.2.1 contains a TSA, thus the condition is dropped. As no restricting condition remains in the sub-query, all the three tuples of join attribute, f, for T1 will be selected. The unique join-value set of attribute f is $\{S01, S02\}$, and it is transmitted to the relevant nodes participating in the join operation. In this example, the set is not sent to node B because no T2 data is stored at node B.

The condition clause of Sub-query B.2.2 received from node B does not contain a TSA and the evaluation of the query shows that tuples 1 and 4 of base table T2 at node A satisfy the query. Thus, the unique join-value set of attribute, f, of T2 becomes {S01, S04}. It is transmitted to node B where a fragment of base table T1 is stored.

Note that there is actually no need to send any unique join-value set to query node because no data will be replicated from itself.

At Node B: Similarly, sub-query B.2.1 is evaluated and a unique join-value set, {S03}, is generated and transmitted to A. Note that the restricting attribute, c, is a TSA at node B and would not be dropped from the sub-query as the case at node A.

At each node, the unique join-value sets are combined, join operation is evaluated, and the retrieval matrices are marked for the data to be replicated.

Thus the TIF created at node A for the base table T1 at node B is as follows:

	a	b	c	d	e
tuple 1	002	Doris	2000	filler	filler
tuple 2	005	Gray	800	600	B

Note that attributes d and e of tuple 1 are occupied by 'fillers' because the values of these two data items are not replicated in the local TIF. A filler is a predefined value for non-replicated cells in the TIF's.

Node B: A TIF created at node B for the base table T1 at node A is as follows:

	a	b	c	d	e	f
tuple 1	001	Acme	filler	filler	filler	S01
tuple 2	003	Exso	filler	filler	filler	S01
tuple 3	004	Giant	filler	filler	filler	filler

TIF created at node B for the base table T2 at node A is as follows:

	f	g	h
tuple 1	S01	Cecil	F

C. Query Processing in TIF Environment

Query processing with TIF's is different from that in a non-replicated environment. Local base tables and TIF's may contain *some* of but not *all* of the data needed to answer a query. To satisfy the query request, it is important to accurately determine the portion of data that satisfies the query but resides at a remote node. Hence, each query is processed at the relevant base table nodes to determine the data items (not available in local base table or in TIF) to be transmitted to the query node. Processing of modification and retrieval queries is presented below.

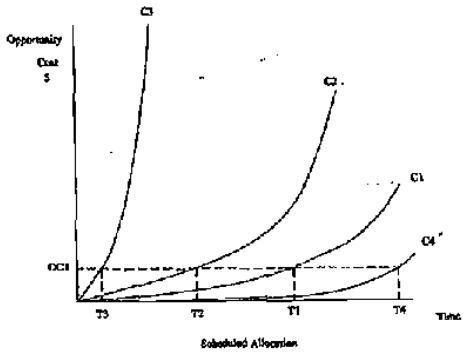


Fig. 4. Scheduling the allocation of TIF's.

Modification Queries: A modification query in a TIF environment is processed in the following manner:

Step 1. Consult data directory to determine the nodes where the relevant base tables are stored. Submit the query to those nodes.

Step 2. Update the relevant base tables. Examine TSA and Retrieval matrices for the base tables to explore whether the updated data items have been replicated at other nodes. Propagate the update to those TIF's that contain a copy of the updated data items.

Step 3. Replace the old TIF data items at the destination nodes.

Retrieval Queries: Single-table retrieval queries and two-way join retrieval queries require different treatments.

Single-Table Retrievals: The processing of a retrieval query is illustrated below:

Step 1. Consult data directory to determine the nodes where the relevant base tables are stored. Submit the retrieval query to those nodes.

Step 2. Process the query at the destination nodes. Examine the Retrieval matrices to determine if the data items exist in the TIF's at the query node. Transmit those data items have not been replicated to the query node.

Step 3. Merge data retrieved from local base table, local TIF, and remote base tables.

Two-Way Join Retrievals: The processing of a two-way retrieval query is illustrated below:

Step 1. Split the two-way join query into two single-table sub-queries. Consult data directory to determine the location of the relevant base tables. Send sub-queries to those nodes.

Step 2. Evaluate queries at destination nodes for the join attribute and transmit the unique join-value set to the nodes which receive the other sub-query.

Step 3. Process the sub-queries and check the respective Retrieval matrix (or matrices, if the node contains both base tables to be joined) at each node where the join values are available. Determine whether the data items that satisfy the join query have been replicated in the query node TIF's. Transmit the data items to the query node if they have not been replicated.

Step 4. Merge the data retrieved from local base table, local TIF, and remote base tables.

D. Scheduling the Allocation and Update of TIF's

Allocation of TIF's to the user nodes may require intense data communication. Therefore, by sending TIF's selectively and/or during the non-peak hours can substantially reduce the communication costs. In situations where the update of critical data items is intense, the tolerance to the delay in update can be very low and cost of inaccuracy of information can be high. By contrast, in time-irrelevant applications where changes in data values are not critical to decision making, the tolerance to non-current data can be high, and the cost of inaccuracy of data can be low. The MLTIF algorithm is based on the trade-off between the opportunity cost due to inaccurate data and cost of maintaining the accuracy.

The opportunity cost of inaccurate data can be expected to increase over time as shown by the curve *C1* in Fig. 4. The *slope of the curve increases as time* increases. Applications having low tolerance to non-current data will experience a cost curve similar to *C2*, where the slope of the curve increases very fast. In an extreme case, where the currency of data is extremely critical, the slope of the cost curve will be infinite as shown by the curve, *C3*. The applications that can tolerate non-current data will have their cost curves similar to *C4*, where the slope of the curve has a long *flat lead time* before deviating from the time axis. Therefore, for a given level of opportunity cost, OC1, MLTIF will schedule the creation and allocation of TIF's every *T1*, *T2*, *T3*, and *T4* time periods for the above-mentioned classes of applications. In addition,

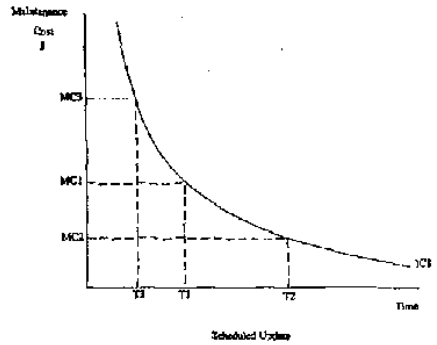


Fig. 5. Scheduling selective updates of TIF's.

MLTIF continuously monitors and adjusts these time periods to reflect the most current usage pattern.

For scheduling selective updates, it is important for MLTIF to know the age of TIF's. The age of a TIF is determined by the elapsed time between the creation of the TIF and the first update of any of its data item in the source base table. As shown in Fig. 5, to keep the cost of non-current data under a specified level, IC1, the age of a TIF data item has to be maintained within T1 units of time. The corresponding cost of currency maintenance is MC1. The currency maintenance cost of time invariant fragments climbs as the age limitations are stricter to reduce the cost of inaccuracy of the information. Therefore, a trade-off between the currency of data and the cost of maintaining it has to be made by MLTIF in the selection of the data items to be stored in the local TIF's.

MLTIF determines the most appropriate schedule for update of TIF's, also by aquiring knowledge from the query history. In making its update decision, MLTIF considers three important issues—'when' to update, 'where' the update source is, and 'how' the update should be done. Refreshing of TIF's can proceed immediately after the update of the base relation or be deferred until a query is made to the TIF. The deferred strategy may include periodic update, update on-demand regardless of queries, random, or a combination of the above methods [4]. The source (s) of the data to be used in the refresh of a TIF may be from its base relation or other view (s) recently created from this base relation.

E. Benefits of the TIF Approach

a) *Reducing database restructuring costs* User's requirements, needs, and the use of data is not constant in the current volatile business environment. A change in the data usage pattern may result in restructuring of a distributed database at a significant cost. The proposed strategy helps reduce the restructuring cost, because TIF is created from query patterns and automatically adjusts to changes in the environment.

b) *Reducing transmission costs* Overall transmission costs for query processing is reduced by storing TIF's locally. Also, the creation of TIF's is based upon query patterns and the corresponding data can be transmitted to its respective sites during economy or non-peak hours.

c) *Reducing update costs* Multiple copies of TIF do not create the problem of update synchronization because, by definition, TIF's do not change for a given time interval.

d) *Improving response time* Unlike other replication techniques, since TIF's do not change for a given time interval, a complete replication of TIF's is possible to provide more local data, and without unnecessarily creating update problems.

e) *Continuously improving performance* Since TIF is tied to the query history, as the size of query history increases, TIF tends towards optimality.

IV. Evaluation of the MLTIF Approach

To demonstrate the usefulness of the MLTIF approach and the conditions under which it may work best, we compare its performance with that of non-replication, full-replication, and materialized view approaches using simulation. The comparison is based on a given query history for a given time interval. Assuming data storage cost being negligible and unit data transmission costs between any two nodes being equal, the costs for creation of replicas, data retrievals, and modification for each approach are formulated. Detailed assumptions and the values assigned to the parameters can be found in the Appendix. The costs are averaged for 50 simulation runs for each setting involving different percentages of modification queries, sizes of network, and number of queries. Detailed findings are presented below.

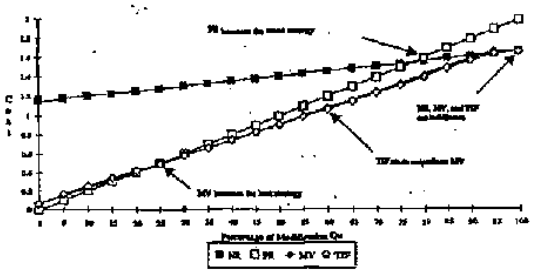


Fig. 6. Costs of the four strategies.(200 base queries, five nodes).

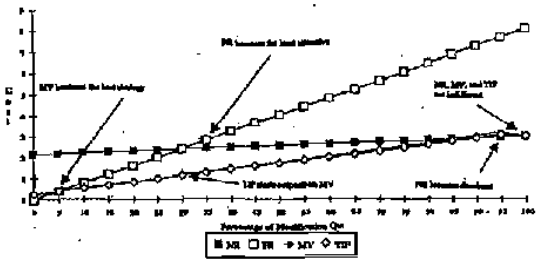


Fig. 7. Costs of the four strategies.(200 base queries, 10 nodes).

A. The Effect of Percentage of Modification Queries

Intuitively, the full-replication approach (FR) should perform best for low modification rates, and the non-replication approach (NR) for high modification rates. Between these two extremes, a partial data replication approach should be more desirable as long as the decrease in data retrieval costs offset the increase in the update synchronization costs. To evaluate TIF's performance relative to the other strategies, we simulate different scenarios varying modification query percentages from 0 to 100 at 5% level. Results of the simulation is summarized in Fig. 6. It is clear that FR has the lowest cost when modification queries in the network is below 25%. As the percentage of modification queries increases, FR becomes less attractive than the two partial data replication approaches as the benefit of full data locality is offset by the high update propagation costs. The materialized view approach (MV) becomes the most attractive alternative when the modification queries varies between 25% and 60%. TIF becomes the dominant approach when modification queries exceeds 60%. The benefit of TIF is that it excludes the time sensitive data from replication and, in the best case, as assumed in this analysis, there is no update propagation. However, TIF requires data transmission to answer queries accessing remote time sensitive data. MV, on the other hand, does not require any data transmission because it materializes these data into local replicas. As such, MV benefits from lower update propagation costs at low modification query percentages. As modification queries increase, MV is overwhelmed with materialized view updates and is surpassed by TIF. Note that when all the queries in the network are modification queries, the costs for NR, MV, and TIF are the same because there is no data replication for MV and TIF. Thus, the three cost lines in Fig. 6 merge when the percentage of modification queries is 100%.

B. The Effect of Network Size

When the size of the network increases, TIF becomes even more attractive. Fig. 7 shows the cost lines of the four strategies when the number of nodes is increased from 5 to 10 with everything else being the same. Here, MV dominates FR at 5% as against 25% modification queries in five-node network. This is because there are more nodes (nine rather than four) requiring the propagation. Since update propagation is more expensive than data retrievals, the benefit of data locality of FR is offset more quickly in a larger network. TIF becomes the preferred strategy at 30% modification queries as against 60% in the five-node network. As more materialized views may be required in a larger network and, consequently, higher cost is incurred in maintaining data currency, NR becomes the

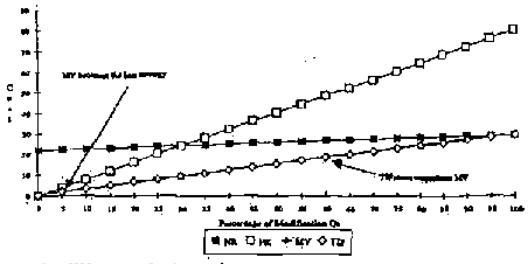


Fig. 8. Costs of the four strategies (2000 base queries, 10 nodes).

most effective strategy when more than 95% are modification queries. Saving in the retrievals of remote time invariant data does not offset the cost of TIF creation as it does in a smaller network.

C. The Effect of Number of Queries

In the above analysis the number of base queries in the network is set to 200. This parameter represents the total number of unique queries in the network. For a 10-node network with 2000 base queries the costs of the four strategies are plotted in Fig. 8. We observe that TIF would not become the preferred strategy until modification queries reaches 60%, which is higher than that in the 200 base queries network (30%). The increase in the number of retrieval queries requires higher retrieval costs of time sensitive data, and, consequently, delays the low cost lead of TIF. NR is never attractive unless all the queries are modification queries. The increases in relative magnitude of both retrieval and modification costs to TIF creation cost makes NR more difficult to surpass TIF. The above reasoning shows that:

1) FR is still the best strategy for small and/or non-modification-query networks,
2) MV has a larger dominance area,
3) TIF also has a larger dominance area and a flatter dominance line, and
4) NR has a much flatter dominance line and does not become attractive until at very high modification percentages and the network size exceeding 100 nodes.

V. Conclusion

This paper presents a learning based approach to the creation and allocation of time invariant fragments. For an application session, first, the time-invariant fragments at a node are defined through conceptual aggregation of expressions in the query history. Next, a set of queries to retrieve remote data for the creation of time invariant fragments are generated. Finally, these remote data retrieval queries are executed to build time invariant fragments and transmitted to the destination node.

To demonstrate the usefulness of the TIF approach and *the conditions under which it may work best, we compare* its performance with that of non-replication, full-replication, and materialized view approaches using simulation. The initial results are promising. It shows that TIF approach can be effective under the following conditions:

1) The percentage of TSA's is low, i.e., most of the data is time invariant;
2) The percentage of modification queries is high, and
3) The size of the network is large.

The current research can be extended in the following directions:

1) *Improving the precision of time-sensitivity definition.* The definition of time-sensitivity is done to attribute level. A more precise definition such as at the value range level could provide more data to time invariant fragments and improve the percentage of local processing.
2) *Refreshing TIF's.* TIF's are created independently for each application session. Time invariant fragments across sessions may overlap, i.e. part of data in the time invariant fragments created in one session is still valid and can be used in another session (a). The detection of these common time-invariant data will reduce data transmission cost.

Appendix

Notation, Formulation, and Values of Parameters for the Costs of the Four Data Allocation/Replication Approaches

Notation. The following notations are used in formulating the costs of the four approaches:

N Number of nodes in the network.

Q_n Total number of base queries in the network. These are unique queries inquired in the network. Additional retrieval queries are added when the network expands and the volume for each node is calculated through a distribution factor, as will be explained later.

P_{mq} Percentage of modification queries in the network.

Q_{mi} Number of modification queries at node i, $\sum_{i=1}^{N} Q_{mi} = Q_n P_{mq}$.

Q_{ri} Number of retrieval queries at node i.

Q_i Number of queries at node i, $Q_i = Q_{ri} + Q_{mi}$.

R_r Replication factor of network retrieval queries per additional node. When a node is broken into multiple nodes (e.g., West Coast is broken into Washington, Oregon, Northern California, and Southern California), a query in the original topology can be requested by one or more nodes in the new network because the nodes sharing the query result in the old network can now inquire themselves. The number of such additional queries is R_r percent of the network base retrieval queries for each additional node. Hence, the number of retrieval queries $= Q_N'(1 - P_{mq}) * [1 + (N - 1)*R_r], N > 1$. Since the number of retrieval queries increases as the network expands, the number of queries in the network also increases.

D_{mi} Distribution factor of network modification queries to node i. That is, $Q_{mi} = Q_n * P_{mq}/N' D_{mi}$. Thus, $\sum_{i=1}^{N} Q_{mi} = \sum_{i=1}^{N} Q_n * P_{mq}/N' D_{mi} = Q_n * P_{mq}$ because the number of modification queries in the network is assumed to be a constant. Therefore, $\sum_{i=1}^{N} D_{mi} = N$.

D_{ri} Distribution factor of network retrieval queries to node i. I.e., $Q_{ri} = Q_N'(1 - P_{mq})' [1 + (N - 1)*R_r]/N' D_{ri}$, where $N > 1$, $\sum_{i=1}^{N} D_{ri} = N$.

P_{mi} Percentage of modification queries at node i. $P_{mi} = Q_{mi}/Q_i$.

A_{ri} Average percentage of retrieval query data is remote for node i's queries.

A_{mi} Average percentage of modification query data is remote for node i queries.

T_r Threshold network size for retrieval queries. When the network is small, it is easier to achieve high data locality through database partition. As the network size increases, the maintenance of high data locality becomes more difficult. When the expansion of the network exceeds a threshold size, the locality of data is sharply reduced to a very low level because the size of the unpartitioned base table is fixed and too much partitioning will result in meaningless fragments. The larger T_r is the larger the network can expand without a sharp reduce of data locality. To capture this nature of data locality, we propose that

$$A_{ri} = \begin{cases} 1) [(N-1)/N]' 1.1^{(N-T_r)/2}, & \text{if } N < T_r \\ 2) [(N-1)/N]' [1 - e^{(T_r - N)/2}], & \text{if } N \geq 3T_r. \end{cases}$$

When $N = 1$, $A_{ri} = 0$.

T_m Threshold network size for modification queries. It is similar to T_r except it is for modification queries. Thus,

$$A_{mi} = \begin{cases} 1) [(N-1)/N]' 1.1^{(N-T_m)/2}, & \text{if } N < T_m \\ 2) [(N-1)/N]' [1 - e^{(T_m - N)/2}], & \text{if } N \geq T_m. \end{cases}$$

When $N = 1$, $A_{mi} = 0$.

P_{tsai} Percentage of node i retrieval queries retrieves TSA's.

A_{tsai} Average coverage of remote TSA's by the retrieval queries of node i.

M_m Cost multiplier for modification queries. It is more costly to modify a data item than to retrieve it because of additional operations and data transmissions. This overhead includes verification of the appropriateness of the modification, transmission of modified data back to the remote nodes from modification originating node, locking of the multiple copies if necessary, verification message of the modification from remote nodes, etc., are required for modification queries. Thus, $M_m > 1$.

R_{MV} Efficacy factor of a materialized view refresh method such as differential files, etc. It represents the ratio of entire-view-refresh cost incurred when a refresh method is applied. The lower the value, the more cost effective the refresh method is.

R_{FR} Efficacy factor of a full-replication update propagation method. The lower the value, the more cost effective the propagation method is.

E_{ri} Expansion factor from average data retrieved per query to overall coverage of data retrieved by queries of node i. A small value of E_{ri} indicates that the retrieval queries at node i access large amount of common remote data.

E_{mi} Expansion factor from remote modification data to overall modified data, both local and remote, by queries of node i. A small value of E_{mi} indicates that a large portion of data modified by the queries at node i are stored at local base tables.

P_{mvm} Percentage of network modification queries modify data materialized in the views in the network.

P_{mvmi} Percentage of network modification queries modify data materialized in the view at node i. The number of modification queries is $Q_n * P_{mq} * P_{mvmi}$.

M_i Percentage of P_{mvm} allocated to node i. Thus, $P_{mvmi} = P_{mvm} * M_i$.

C_{NR} Total costs of non-replication strategy.

C_{FR} Total costs of full-replication strategy.

C_{MV} Total costs of materialized view strategy.

C_{TIF} Total costs of TIF strategy.

Major assumptions To simplify the complexity of the problem, we assume the following:

1) Data storage/maintenance cost is negligible.
2) Unit data transmission costs between any two nodes are equal.
3) Database directory is replicated at all the nodes, i.e., the *possible* location of data can be identified locally.
4) Messages for requesting remote data in retrieval queries are included in the cost of the transmissions of the requested data. The messages for modifying remote data are also included in the remote modification cost. Typical messages include requesting for locks at remote nodes, granting locks from remote nodes, and requesting for releasing locks at remote nodes. To capture the relative higher cost of modification to retrieval queries for the same data, we include a cost multiplier for modification queries. Its values are assumed to be 1.5 in this analysis. The value means a modification query is 50% more costly than a retrieval query to access the same amount of remote data. The effect of the communication cost of these messages in the design of distributed systems has

been studied.[20]

5) Additional costs for unsuccessful retrieval and modification queries are assumed to be zero.
6) The amount of data transmitted in refreshing a materialized view is lower than that of creating the entire view. Whenever a discrepancy between the data in the view and that in the base table occurs (because of a modification query), only a portion of the view is refreshed. Various materialized view refresh techniques have been proposed and the effectiveness of these techniques depends upon environments under which they are utilized. To capture the performance of different materialized view refresh techniques, we include an efficacy factor in our model. In the current analysis, we assume a value of 10%, which indicates that on average only 10% of the view is refreshed to maintain the desired level of currency of data, although this may depend upon the refresh strategy selected.
7) The amount of data transmission in maintaining database currency for full-replication strategy is not proportional to the number of nodes. As for materialized views, more effective update propagation techniques could be utilized to reduce currency maintenance cost. To capture this phenomenon, we include an efficacy factor in our model similar to that for materialized views. We assume a 10% value for the factor in the current analysis. The value indicates that on average only 10% of the modified data are transmitted to other nodes to maintain database currency, although this may vary from system to system.
8) The value of a parameter is the same for all the nodes. Heterogeneous network nodes are expected in practice. But to keep the analysis from being too complicated, we assume the same value for a parameter across the network.

Cost formulation. The costs of the four strategies are formulated as follows:

Non-Replication

$$C_{NR} = \sum_{i=1}^{N}[A_{sri} * Q_i * (1 - P_{mi})] \quad \text{(Retrieval Cost)}$$

$$+ M_m * \sum_{i=1}^{N}[A_{mri} * Q_i * P_{mi}] \quad \text{(Modification Cost)}$$

Full-Replication

$$C_{FR} = M_m * (N-1) * R_{FH} * \sum_{i=1}^{N}[E_{mi} * A_{mri} * Q_i * P_{mi}] \quad \text{(Modification Cost)}$$

Materialized Views

$$C_{MV} = \sum_{i=1}^{N}[E_{ri} * A_{mi}] \quad \text{(Creation Cost)}$$

$$+ M_m * \sum_{i=1}^{N}[A_{mri} * Q_i * P_{mi}] \quad \text{(Modification Cost)}$$

$$+ R_{MV} * M_m * [E_{ri} * A_{rri} * Q_{ri} * P_{mvq} * P_{mvmi}] \quad \text{(Refresh Cost)}$$

Time-Invariant Fragmentation

$$C_{TIF} = \sum_{i=1}^{N}[E_{ri} * A_{ri} * (1 - A_{rtsai})] \quad \text{(Creation Cost)}$$

$$+ \sum_{i=1}^{N}[A_{rri} * A_{rtsai} * Q_i * (1 - P_{ri}) * P_{rtsai}] \quad \text{(TSA's Retrieval Cost)}$$

$$+ M_m * \sum_{i=1}^{N}[A_{mri} * Q_i * P_{mi}] \text{[Modification Cost)}$$

Parameter Values. The parameter values used in the current study are based on realistic business scenarios. They are as follows:

A_{rtsai} random number between 0% and 100%, with 80% probability of being below 20%, 15% probability of being between 20% and 50%, and 5% probability of being above 50%
D_{ri} 1
D_{ti} 1
E_{ri} 2.5
E_{mi} 3
M_i 10%
M_m 1.5%
P_{mvmi} 10%
P_{rtsai} random number between 0% and 100%, with 80% probability of being below 20%, 15% probability of being between 20% and 50% and 5% probability of being above 50%
R_r 1%
R_{MV} 10%
R_{FH} 10%
T_r 50
T_m 50

REFERENCES

[1] M. E. Adiba and B. G. Lindsay, "Database snapshots," in *Proc. Int. Conf. VLDB*, Montreal, PQ, Canada, 1980, pp. 86–91.
[2] R. Alonso, D. Barbara, H. Garcia-Molina, and S. Abad, "Quasi-copies: Efficient data sharing for information retrieval systems," in *Lecture Notes in Computer Science*, vol. 303, J. W. Schmidt, S. Ceri, and M. Missikoff, ed. Springer-Verlag, 1988, pp. 443–468.
[3] P. M. G. Apers, "Data allocation in distributed database systems," *ACM TODS*, vol. 13, no. 3, pp. 263–304, 1988.
[4] J. Blakeley, P. Larson, and F. Tompa, "Efficiently updating materialized views," *ACM SIGMOD 86*, pp. 61–71, 1986.
[5] R. G. Casey, "Allocation of copies of a file in an information network," in *Proc. Spring Joint Computer Conf., AFIPS*, 1972.
[6] S. Ceri, S. Navathe, and G. Wiederhold, "Distribution design of logical database schemas," *IEEE Trans. Software Eng.*, pp. 487–504, 1983.
[7] S. Ceri, B. Pernici, and G. Wiederhold, "Optimization problems and solution methods in the design of data distribution," *Information Systems*, vol. 14, no. 3, pp. 261–272, 1989.

[8] A. Chaturvedi, C. Hutchinson, and D. Nazareth, "A synergistic approach to manufacturing systems control using machine learning," *J. Intelligent Manufacturing*, vol. 3, pp. 43–57, 1992.
[9] W. W. Chu, "Optimal file allocation in a multiple computer system," *IEEE Trans. Computers*, vol. C-18, 1969.
[10] L. W. Dowdy and D. V. Foster, "Comparative models of the file assignment problem," *ACM Computing Surveys*, vol. 14, no. 2, pp. 287–313, 1982.
[11] A. Dutta, "Modeling of multiple copy update costs for file allocation in distributed databases," *Int. J. Computer and Info. Sci.*, vol. 14, no. 1, pp. 29–54, 1985.
[12] E. R. Hansen, "A performance analysis of view materialization strategies," in *Proc. ACM-SIGMOD Int. Conf. Manage. of Data*, 1987, pp. 440–453.
[13] B. Kahler and O. Risnes, "Extending logging for database snapshot refresh," in *Proc. Int. Conf. VLDB*, 1987, pp. 389–398.
[14] B. Lindsay, L. Haas, C. Mohan, H. Pirahesh, and P. Wilms, "A snapshot differential refresh algorithm," in *Proc. ACM-SIGMOD Int. Conf. Manage of Data*, 1986, pp. 53–60.
[15] S. Navathe, S. Ceri, G. Wiederhold, and J. Dou, "Vertical partitioning algorithms for database design," *ACM TODS*, vol. 9, no. 4, pp. 680–710, 1984.
[16] S. Navathe and M. Ra, "Vertical partitioning for database design: A graphical algorithm," in *Proc. ACM-SIGMOD Int. Conf. Manage. of Data*, 1989, pp. 440–450.
[17] T. R. Rakes, L. S. Franz, and A. Sen, "A heuristic approximation for reducing problem size in network file allocation models," *Comput. & Opn. Res.*, vol. 11, no. 4, pp. 387–395, 1984.
[18] D. Sacca and G. Wiederhold, "Database partitioning in a cluster of processors," *Proc. Int. Conf. VLDB*, 1983, pp. 242–247.
[19] A. Segev and W. Fang, "Optimal update policies for distributed materialized views," *Manage. Sci.*, accepted for publication.
[20] A. Segev and J. Park, "Updating distributed materialized views," *IEEE Trans. Knowledge and Data Eng.*, vol. 1, pp. 173–84, 1989.
[21] D. G. Shin and K. B. Irani, "Partitioning a relational database horizontally using a knowledge-based approach," *ACM SIGMOD Record*, vol. 14, no. 4, pp. 95–105, 1985.

Alok R. Chaturvedi, for a photograph and a biography, please see page 193 of this issue of this TRANSACTIONS.

Ashok K. Choubey received the M. S. degree in computer engineering from Syracuse University in 1987.

He is a Senior Analyst with Information Systems—Data Management Applications Group, Bell Atlantic. He has extensive experience in distributed systems design, client/server applications, and database administration. His current research interests include enterprise modeling and cooperative processing.

Andrew Kuo received the M. S. degree from the University of Missouri-St. Louis, and the Ph. D. degree from Purdue University, West Lafayette, IN, both in management information systems. Currently, he is an Associate Professor of Business Administration at the national Chung Cheng University Taiwan. His prior experience includes information systems planning, development, and implementation. His research interests include distributed database, network design, machine learning, and information systems development.

Parallel Hierarchical Evaluation of Transitive Closure Queries

Maurice A.W. Houtsma*
Department of Computer Science, University of Twente (email: houtsma@cs.utwente.nl)

Filippo Cacace, Stefano Ceri†
Dipartimento di Elettronica, Politecnico di Milano (email: cacace, ceri @ipmail.polimi.it)

Abstract

This paper presents a new approach to parallel computation of transitive closure queries using a semantic data fragmentation. Tuples of a large base relation denote edges in a graph, which models a transportation network. We propose a fragmentation algorithm which produces a partitioning of the base relation into several fragments such that any fragment corresponds to a subgraph. One fragment, called high-speed fragment, collects all edges which guarantee maximum speed, these edges correspond to highways or to high-speed intercity trains. Thus, the fragmentation algorithm induces a hierarchical relationship between the high-speed fragment and all other fragments. With this fragmentation, any query about paths connecting two nodes can be answered by using just the fragments in which nodes are located and the high-speed fragment. In general, if each fragment is managed by a distinguished processor, then the query can be answered by three processors working in parallel. This schema can be applied recursively to generate an arbitrary number of hierarchical levels.

1 Introduction

Over the past few years, deductive databases have emerged to bridge the gap between data- and knowledge-base systems. Relational databases are capable of efficiently handling large amounts of data on secondary storage, but their interface is sometimes considered as rather awkward and their expressive power is limited. Therefore, deductive databases offer a logic-based interface, called Datalog, that enables easy formulation of complex queries and, more importante enables formulation of recursive queries on top of a relational system. This has triggered a vast body of research on optimization strategies for recursive queries, both in an algebraic and in a logic context (see, e.g., [4]). A simple, but very important type of recursion is the transitive closure operation [11]. Transitive closure operations in algebra amount to the class of linear sirups in Datalog [3], which class has attracted most research on optimization. An example of a transitive closure query, combined with an aggregate computation, is the bill-of-material problem: finding all transitive components of a given part.

An important feature of database technology of the nineties is the use of parallelism for speeding up the execution of complex queries. In particular, *intra-query parallelism* enables the distribution of complex queries to multiple processors. Fragmentation is essential to intra-query parallelism, as it enables a very natural partitioning of query processing. Each processor controls a disk which stores fragments of relations; with this architecture, it is possible to execute selections, projections, and some joins in a distributed way on each processor, and then collect from each processor the result of these operations. Today, a number of *research prototypes and a few commercial systems* support fragmentation parallelism (e.g., [12, 13]).

The new class of recursive queries can, by its regular structure and complexity of operations, *extremely* benefit from intra-query parallelism. Therefore, research is now being conducted on parallel execution of recursive queries, both in a logic and in an algebraic context [5, 7, 15]. Some research focuses on the use of hash-based fragmentation [5, 14]; some logic-based approaches focus instead on assigning Datalog rules to processors [10]. An overview of current research on parallel execution strategies for transitive closure is given in [3].

In this paper we generalize the "disconnection set" approach introduced in [7, 8]. In the disconnection

*The research of Maurice Houtsma has been made possible by a fellowship of the Royal Netherlandst Academy of Arts and Sciences

†Filippo Cacace and Stefano Ceri are supported by the Esprit project STRETCH and by the CNR project LOCIDATA+

0-8186-2295-4/91 $01.00 © 1991 IEEE

set approach, the semantics of the application domain is used to achieve a significant computation speedup for several types of transitive closure queries: graph reachability, shortest path, and bill-of-material. In this paper we concentrate on the shortest path problem, and we extend and improve on the disconnection set approach. The approach of this paper, called *parallel hierarchical evaluation*, uses a new fragmentation schema that partitions a base relation into several fragments; one of them, called *high-speed fragment*, has special properties. This fragmentation schema allows us to direct queries to specific fragments (in general, 3 fragments are sufficient to solve any shortest path query); this means that relevant data to answer a query are pre-selected. At the same time, each query can be answered in parallel by the processors associated to the relevant fragments; in particular, when each fragment is controlled by one processor, then a query is naturally executed in parallel on three processors.

Our fragmentation algorithm turns out to be familiar to one of the heuristics proposed in [1]. There the issue was to divide a relation into domains in such a way that the search space can be pruned dynamically. However, they do not consider distributed computation; instead they focus on storing precomputed information (in the order of the size of the relation) about the shortest connection between any pair of nodes in a fragment. This leads to a considerable overhead compared to disconnection sets [7]. The assumption made in [1] that domains do not overlap is either strict and not likely in general; overlapping domains would in their approach lead to a significant effort in trying to bound the search. Although our approach has some characteristics in common with [1], there are significant differences. We give a proper definition of fragments and a full description of a fragmentation algorithm, concentrating on the use of distributed computation for all sorts of transitive closure queries (including e.g. bill-of-material).

The structure of this paper is as follows. In Sec. 2 we give a short description of the disconnection set approach. In Sec. 3 we describe the generalisation of this approach to parallel hierarchical evaluation. In Sec. 4 we present an algorithm for fragmentation design[1]. In Sec. 5 we discuss query processing, and in Sec. 6 update management. Finally, in Sec. 7 we draw some conclusions and discuss issues for further research.

[1] Although a number of strategies use data fragmentation to achieve parallel execution [5, 14, 15], they typically assume fragmentation based on basic functions, without discussing how to achieve a good fragmentation design.

2 A primer on disconnection sets

The disconnection set approach, described in [7, 8], uses a fragmentation that is especially tailored to parallel execution of recursive queries. It was suggested by real-world observations concerning travel problems. The basic idea underlying the disconnection set approach is very simple and can be illustrated by considering a railway network in Europe.

Assume that connections in the European railway network are naturally fragmented by nation, and that each fragment is stored on geographically distributed computers that can be accessed through a distributed database system. To find the shortest path from Paris in France to Milano in Italy we may split the question in a number of separate subqueries: find a path from Paris through France to the north-eastern border with Switzerland, then through Switzerland to the Italian border, and finally through Italy to Milano.

This fragmentation leads to a highly selective search process, consisting of determining the properties of connections from the origin to the first border, then between borders of the intermediate fragment, and finally from the last border to the destination city. These queries have the same structure; they apply only to a fragment of the database, and can be executed in parallel.

The relational representation of this is as follows. The connection information is stored into a relation R; each tuple corresponds to an arc of the graph G, which can have cycles. By effect of the fragmentation, R is partitioned into n fragments R_i, $1 \leq i \leq n$, each stored at a different computer or processor. This fragmentation induces a partitioning of G into n subgraphs G_i, $1 \leq i \leq n$. *Disconnection sets* DS_{ij} are given by $G_i \cap G_j$.

In order to process queries independently on the fragments, it is required to store some *complementary information* about the identity of border cities (i.e., the nodes in the disconnection set) and the properties of their connections; these properties depend on the particular recursive problem considered. For instance, for the shortest path problem it is required to precompute the shortest path among any two cities on the border between two fragments. Such shortest paths may cross the border many times and, therefore, have to be computed and maintained based on the entire graph G; this is discussed in [7]. Complementary information about DS_{ij} is stored together with both fragments G_i and G_j.

In the disconnection set approach, we assume that disconnection sets are much smaller than the fragments, and that they do not overlap (which will in

general be the case). Given these assumptions, the disconnection set approach leads to an effective use of parallel computation on n processors (where n is the number of fragments involved) [7]. This is especially true for fragmentations that are *loosely connected*, i.e., whose graph of components is acyclic; the graph of components has one node for each fragment and one edge for each nonempty disconnection set.

Proofs on the correctness of the disconnection set approach for the various types of transitive closure queries can be found in [8].

3 Parallel Hierarchical Evaluation

In the disconnection set approach, all fragments are "semantically equal." The parallel hierarchical evaluation generalises the disconnection set approach, by making some of the fragments more equal than others. This distinction is based on real-life observations concerning transport problems.

3.1 Informal Description

If we consider the railway network of many European countries, we note that the countries are subdivided into geographical regions. In each region, a number of slow trains stop at every station; remote regions are connected by inter-city trains that stop only at a few stations, typically the major cities of a region. For long-distance travel, one typically uses the regional network around the departure city in order to reach the high-speed network of inter-city trains, then uses the inter-city trains, and finally uses the regional network around the destination city to arrive at the destination. If arrival and destination are in adjacent regions, an exception to this rule is possible: it might be better to use the slow trains that connect the two regional networks, instead of using the high-speed inter-city trains.

The parallel hierarchical evaluation exactly mimics this intuitive approach to travelling. A *small* subset of the connections is declared to be high-speed; these connections are stored as a separate fragment H. The remaining connections are partitioned into n fragments (corresponding to geographical regions). Each fragment is stored on a separate processor, together with the complementary information for each of its disconnection sets. H is stored on a separate computer, together with the complementary information about the disconnection sets between H and the various other fragments. As in the disconnection set approach, we assume that disconnection sets are small compared to the size of fragments.

Any two fragments are declared as either adjacent or nonadjacent. Two adjacent fragments have a nonempty disconnection set. We build a fragmentation so that the shortest path between any two nodes belonging to fragments G_i and G_j is included within these fragments and within H, but does not include edges *from other fragments*. In particular, if G_i and G_j are nonadjacent, then the shortest path must include some edge from H.

Note that for achieving a good balance of the work and a profitable parallel computation, it is required that H be small (because it is used in most computations), the fragments be approximately of similar size (even workload), and the disconnection sets be small (to minimise overhead and precomputation).

3.2 Formal Description

We now give a formal description of parallel hierarchical evaluation in terms of the underlying graph and its fragmentation. First we introduce a graph G, its subgraphs G_i, and the high-speed fragment H; DS_{ij} denotes the disconnection set between two generic fragments G_i and G_j, DSH_i denotes the disconnection set between G_i and H. The function W_G is a weight function that assigns a weight to each edge of G.

$$G = (V, E) \qquad W_G : E(G) \rightarrow I\!N$$

$$G_1 = (V_1, E_1) \quad \ldots \quad G_n = (V_n, E_n) \quad H = (V', HS)$$

$$V_1 \cup \ldots \cup V_n \cup V' = V \qquad E_1 \cup \ldots \cup E_n \cup HS = E$$

$$\forall i, j: \quad E_i \cap E_j = \emptyset \qquad \forall i: \quad E_i \cap HS = \emptyset$$

$$DS_{ij} = V_i \cap V_j \qquad DSH_i = V_i \cap V'$$

We assume a function $sh(v_i, v_j)$ that returns the set of arcs constituting the shortest path in G between nodes v_i and v_j. Whenever we mention the shortest path we mean the path for which the summation of the weight of the edges is minimal. We like to design a data fragmentation which satisfies the following property:

Property 3.1 *The shortest path between any two nodes contained in fragments G_k and G_k includes only edges from G_k, G_k, and the high-speed fragment H:*

$$\forall v_i \in V_k, v_j \in V_k, sh(v_i, v_j) \subseteq E_k \cup E_k \cup HS$$

We now introduce the notions of adjacency and non-adjacency.

adjacency Two fragments G_h and G_k, with $h \neq k$, are adjacent if the following properties hold:

1. $\forall v_i \in V_h, v_j \in V_k : sh(v_i, v_j) \subseteq E_h \cup E_k \cup HS$
2. $V_h \cap V_k \neq \emptyset$

non-adjacency Two fragments G_h and G_k, with $h \neq k$, are non-adjacent if the following properties hold:

1. $\forall v_i \in V_h, v_j \in V_k : sh(v_i, v_j) \subseteq E_h \cup E_k \cup HS$
2. $V_h \cap V_k = \emptyset$, with $h \neq k$

The property that enables parallel hierarchical evaluation is the following:

Property 3.2 *Any two pair of fragments G_h, G_k, with $h \neq k$, are either adjacent or non-adjacent.*

Note that Property 3.1 is a logical consequence of Property 3.2, as it is implied by condition 1, common to both adjacency and nonadjacency conditions.

If we postulate that each fragment is managed by a separate processor and is stored together with the *complementary information of all its disconnection* sets, we may then formulate the following theorem.

Theorem 3.1 *If Property 3.1 holds then the shortest path between any two internal nodes v_i and v_j of fragments G_i and G_j can be computed on three processors in parallel.*

Note that if either of the departure or arrival node fall into the disconnection set between two fragments, then the node(s) must be interpreted as belonging to both fragments; thus, in the worst case (when both departure and arrival nodes are not internal), four of the above computations must be performed; the shortest path is obtained as the minimum of the shortest paths obtained from each computation. Due to our assumptions on the size of disconnection sets, this case is unlikely.

Also note that this formalization allows for trivial solutions, such as declaring the complete graph to be in HS. Obviously, this is not what we want if we aim to achieve parallel computation. When designing the fragmentation we shall generate at least n fragments (with n reasonable large), where the fragments are approximately equal in size.

4 Fragmentation design

In this section we describe how to design a fragmentation that satisfies Property 3.1; fragmentation design is a hard problem, and this section constitutes the core of this paper. Due to space considerations, we only describe the algorithm informally here, a full version is given in [3] *The design is initiated by the user's* choice of "region centers": the user pre-determines the number of fragments n, and chooses for each fragment the node c_i of the graph which is situated approximately in its center. This initial choice will influence the final outcome and is heuristic in nature. From this choice on, the algorithm develops a fragmentation that satisfies the requirements described at the end of the previous section. Let C denote the set of fragment centers. Fragmentation design is then conducted in five steps.

Step 1. During this step, the shortest path—in terms of the sum of the weight of the edges—between any pair of the nodes in C is computed. All edges belonging to these paths are removed from the graph G and put into the high-speed fragment. Note that the connections in the high-speed fragment need not be acyclic.

Step 2. Then, edges of G are progressively assigned to fragments. This is done by starting from center nodes and by progressively including neighbour nodes and the edges leading to them into the fragments. At each iteration, the next node which is included into a fragment is the unassigned node with minimum distance from a center node—by construction this node can be reached by an edge starting from a node that was already assigned to a fragment. In this way, the fragments being generated have approximately the same diameter (again in terms of the sum of the weight of the edges, not in terms of the number of edges constituting a path). Whenever an edge connects two nodes assigned to different fragments, that edge is marked as "critical" and included into a set D to be examined in step 3. After step 2, each node is assigned to exactly one fragment. The following property holds, where c_i denotes the center of fragment G_i and $sl(v_i, c_i)$ denotes the length of the shortest path between v_i and c_i:

$$\forall i, k : v_k \in V_i, j \neq i, sl(v_k, c_i) \leq sl(v_k, c_j)$$

However, the fundamental Property 3.1 does not hold. As an example of possible property violation, consider Figure 1. Assume that the shortest path between nodes a and d goes through nodes b and c; the shortest path connecting node a of G_1 and d of G_3 thus uses an edge from G_2 and violates Property 3.1. Note that this problem cannot be solved by assigning critical edges (in this case edges (a, b) and (c, d)) to H.

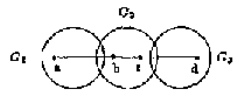


Figure 1: Critical nodes and edges during fragmentation design

Step 3. Step 3 considers "critical edges" in D. We start by building the set B of "boundary nodes" of fragments, defined as the set of nodes from which one critical edge departs:

$$B = \{v_i \mid \exists e_{ij} \in D\}$$

Then, for every pair of nodes v_i and v_j in B, with $v_i \in G_h$ and $v_j \in G_k$, we compute the shortest path that connects them. If this path includes any edge from another fragment G_w ($w \neq h \wedge w \neq k$), then all the edges of this path are inserted into the high-speed fragment H and deleted from D and G_w. After this step, Property 3.1 holds; intuitively, it holds for all pairs of "border nodes" of fragments, by direct construction; but all paths connecting any pair of "interior nodes" have to go through some "border nodes", where they connect to the high-speed fragment, if that is needed.

Although this step is highly combinatorial, its computation is not too hard, since we assume $|B| << |V|$, and we need to consider $fS(\times(|B|-1)/2$ possible combinations. Section 4.1.2 shows optimizations that apply to Step 3.

Step 4 This step determines whether any two fragments are adjacent or nonadjacent. For all pairs of fragments, the number of critical edges in D that connect them is counted. If this number is below a threshold t, then the fragments are defined as nonadjacent and the edges connecting them are put into the high-speed fragment HS. If the number of connecting edges is greater than t, then fragments are defined to be adjacent, and each connecting edge is *arbitrarily assigned to either of them. Eventually,* D becomes empty and all edges are assigned to a fragment; this terminates the fragmentation design.

Note that if t is very high, then many pairs of fragments are defined as *non-adjacent*, and the high-speed fragment is large. Conversely, if t is very low, then many pairs of fragments are defined as adjacent, and the high-speed fragment is small. Query processing is best performed in parallel when most fragments are nonadjacent. The "optimal" situation (i.e., balancing the size of HS, the size of the disconnection sets, and the size of the fragments to achieve a fast response time) for a specific graph may be found by adjusting t.

Step 5. The last step computes the complementary information (see Sec. 2) for the disconnection sets between the adjacent fragments and between each fragment and the high-speed fragment.

Note that this algorithm indeed avoids a trivial solution, and keeps to the guidelines mentioned at the end of Sec. 3.2. In particular, by building fragments where the boundaries are approximately the same distance from the center, and by delaying the assignment of critical edges to fragments, the generated fragments are more or less of equal size.

4.1 Efficient implementation

In this section we discuss the efficient implementation of the hardest steps of the allocation algorithm. Step 4 is computationally easy. Steps 1, 3, and 5 are structurally very similar, as they require solving a large number of shortest path problems. We first address the optimization of Step 2, then we consider Steps 1, 3, and 5 together.

4.1.1 Step 2

In Step 2 of the algorithm we add edges to fragments, based on the minimal distance of nodes to centers. This step is repeated $O(|G|)$ times, and thus must be performed efficiently. Indeed, there is an efficient implementation that greatly reduces its complexity.

We store a list $L1$ of ternary tuples (c_i, v_k, d_k), representing that the shortest path *from center* c_i *to* node v_k has length d_k. $L1$ is sorted based on the third column and represents the partial spanning trees that start from the center nodes in G. We also store a list of edges $L2$ that can be joined to either one center c_i or to one node v_k of $L1$ and have not yet been included into a spanning tree; $L2$ is sorted based on the weight $W_G(\text{tuple})$. Initially, $L1$ is empty and $L2$ includes all tuples of E which survive Step 1 and can be joined with any center c_i, i.e., one of the nodes connected by the edge is a center node.

At each iteration, we search for the shortest connection that can be made between an element of $L1$ and an element of $L2$, to include edges from $L2$ into fragments in increasing order of their distance from the centers. An iteration corresponds to adding one tuple to $L1$, deleting one tuple from $L2$, and possibly moving some tuples from E to $L2$.

be computed by running the Dijkstra algorithm on a separate processor; a version of the algorithm which uses extended relational algebra is discussed in [7].

Once the three subqueries are processed, a post-processing phase is needed to choose the shortest path by taking into consideration the various alternative ways of combining the three parts. Let:

- $P1$ denotes a binary relation storing the result to the first subquery; the first column stores the nodes of DSH_h and the second column stores the distance from v_i to them.
- $P2$ denotes a ternary relation storing the result to the second subquery; the first column stores nodes of DSH_h, the second column stores nodes of DSH_k, and the third column stores the distance between them.
- $P3$ denotes a binary relation storing the result to the third subquery; the first column of $P1$ stores the nodes of DSH_k and the second column stores the distance from them to v_j.

Then, in the following relational expression, each tuple corresponds to a different path traversing disconnection sets in all possible ways:

$$T = ((P1 \bowtie_{1=1} P2) \bowtie_{4=1} P3)$$

The following tuple function can be evaluated, adding column 8 to T: $T.8 := T.2 + T.5 + T.7$. The shortest path corresponds to the minimum value of column T.8. Note that all these operations are here described relationally, but are *really performed in main memory* using arrays as data structures.

5.2 Adjacent fragments

If fragments G_h and G_k are adjacent, we also need to consider paths directly connecting them; these are divided into two independent parts:

- From v_i to DS_{hk};
- From DS_{hk} to v_j.

The first part is computed by using G_h and the complementary information associated to DS_{hk}; the second part requires only G_k. (The complementary information can again be used in the computation on either fragment.) Each subquery can be performed independently, hence two processors may be used.

Post processing is needed to evaluate the best direct shortest path. In order to do so, we denote $P1$ as the binary relation storing the results of the first subquery and $P2$ as the binary relation storing the results of the second subquery. $P1$ and $P2$ are now processed in a similar way as discussed in the previous subsection. Finally, the best direct shortest path is compared with the best shortest path computed by using the high speed fragment, as discussed in Section 5.2.1.

6 Update Management

Unfortunately, hierarchical fragmentation is very sensitive to updates, and therefore is recommended for base relations which are rather stable. An update is *localized within a fragment* if the following two conditions hold:

1. The tuple which is either added or deleted or updated connects two nodes which are internal to the same fragment G_i.
2. This change does not alter the shortest path between any pair of "border nodes" of the fragment, i.e., nodes included in a disconnection set.

Localized updates can be performed on each fragment, without need of recomputing hierarchical fragmentation. Updates which are not localized, however, cannot be dealt with easily[5]. For this reason, we assume that nonlocalized updates should be collected over rather long periods of time, and then applied all together; data distribution should then be re-designed. This update practice is typically used in transportation systems.

7 Conclusions and further research

This paper has presented an approach to the parallel evaluation of shortest path queries on a large base relation. We have discussed the properties that make hierarchical evaluation possible and attractive; we also have discussed how to develop an initial fragmentation that is suited to hierarchical evaluation, how to process queries in parallel, and how to deal with updates. In this paper we have concentrated on the shortest path problem, but our approach can easily be generalised to other problems—such as graph reachability or bill-of-material—by using slightly different complementary information (see [8]).

[5] The apparently easy solution of adding new tuples to the high-speed fragment is unfortunately incorrect; finding a counter-example is an easy exercise, left to the reader.

In the implementation of hierarchical fragmentation, several other optimizations are possible. In particular, the *high-speed* fragment can be *stored in main memory*, and shortest paths of the high-speed fragments can be *pre-computed and permanently stored*.

The hierarchical approach can be generalized to an arbitrary large number of levels. For instance, the high-speed fragment can be further partitioned into one super-high-speed fragment and several high-speed fragments; this is particularly useful if the high-speed fragment is relatively large. Such generalizations correspond to real-life heterogeneous transport systems (for instance, local transportations, trains, and airplanes). In [1]—where distributed computation was not considered, but similar techniques were used for pruning the search space—such a generalization proved useful for two levels, and assuming that the top levels are small compared to the lower level fragments it would also be useful for multi-level fragmentation. This is exactly what we envision to be the case when we fragment the high-speed network using our fragmentation algorithm.

References

[1] Agrawal, R. and Jagadish, H.V. "Efficient search in very large databases," in *Proc. 14th Int. Conf. on Very Large Data Bases*, Los Angeles, 1988, pp. 407–418.

[2] Bancilhon, F., Maier, D., Sagiv, Y., and Ullman, J.D. "Magic sets and other strange ways to implement logic programs," in *Proc. ACM SIGMOD-SIGACT Symp. on Principles of Database Systems*, Cambridge, USA, March 1986, pp. 1–15.

[3] Cacace, F., Ceri, S., and Houtsma, M.A.W. "An overview of parallel strategies for transitive closure on algebraic machines," in *Proc. Workshop on Parallel Database Systems*, Noordwijk, the Netherlands, Sept. 1990; also appeared as Lecture Notes in Computer Science No. 503, Springer-Verlag, pp. 44–62.

[4] Ceri, S., Gottlob, G., and Tanca, L. *Logic programming and databases*, Springer-Verlag 1990.

[5] Cheiney, J.P. and de Maindreville, C. "A parallel strategy for transitive closure using double hash-based clustering," in *Proc. 16th Int. Conf. on Very Large Data Bases*, Brisbane, Australia, Aug. 1990, pp. 347–358.

[6] Ganguly S., Silberschatz A., and Tsur S. "A framework for the parallel processing of Datalog queries," in *Proc. ACM-Sigmod Conference*, Atlantic City, USA, May 1990.

[7] Houtsma, M.A.W., Apers, P.M.G., and Ceri, S. "Distributed transitive closure computations: the disconnection set approach," in *Proc. 16th Int. Conf. on Very Large Data Bases*, Brisbane, Australia, Aug. 1990, pp. 335–346.

[8] Houtsma, M.A.W., Apers, P.M.G., and Ceri, S. "Complex transitive closure queries on a fragmented graph," in *Proc. 3rd Int. Conf. on Database Theory (ICDT'90)*, Lecture Notes in Computer Science, Springer-Verlag, Dec. 1990.

[9] Houtsma, M.A.W., Cacace, F., and Ceri, S. "Parallel Hierarchical Evaluation of Transitive Closure Queries," Technical Report 924, University of Twente, the Netherlands, Dec. 1990.

[10] Hulin, G. "Parallel processing of recursive queries in distributed architectures" in *Proc. 15th Int. Conf. on Very Large Data Bases*, Amsterdam, the Netherlands, 1989, pp. 87–96.

[11] Ioannidis, Y.E. "On the computation of the transitive closure of relational operators," in *Proc. 12th Int. Conf. on Very Large Data Bases*, Kyoto, Japan, Aug. 1986, pp. 403–411.

[12] Kersten, M.L., Apers, P.M.G., Houtsma, M.A.W., van Kuijk, H.J.A., and van de Weg, R.L.W. "A distributed, main-memory database machine," in *Proc. of the 5th Int. Workshop on Database Machines*, Karuizawa, Japan, Oct. 5–8, 1987.

[13] Tandem Database Group "NonStop SQL, a distributed, high-performance, high-availability implementation of SQL," Tandem report, April 1977.

[14] Valduriez, P. and Khoshafian, S. "Parallel Evaluation of the Transitive Closure of a Database Relation," in *Int. Journal of Parallel Programming*, 17:1, Feb. 1988.

[15] Wolfson, O. "Sharing the Load of Logic-program Evaluation," in *Proc. Int. Symp. on Databases in Parallel and Distributed Systems*, Austin, Texas, Dec. 5–7, 1988, pp. 46–55.

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application	:	Kenneth P. Baclawski
Serial No.	:	08/318,252
Filed	:	October 5, 1994
For	:	DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD
Examiner	:	C. Lewis
Attorney's Docket	:	NU-360XX

Group Art Unit: 2307

* *

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: BOX NON-FEE AMENDMENT, Assistant Commissioner for Patents, Washington, D.C. 20231 on 6/7/96.

By _____
Stanley M. Schurgin
Registration No. 20,979
Attorney for Applicant(s)

* *

AMENDMENT

BOX NON-FEE AMENDMENT
Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

In response to the Office Action dated April 16, 1996, please amend the above-identified patent application as follows:

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

In the claims:

Please amend claims 1, 6, 8, 12, 13 and 17 as follows:

Sub B1

1. (Amended) A method for information retrieval using fuzzy queries [of accessing data] in a distributed database system having a plurality of home nodes [node] and a plurality of query nodes connected by a network, said method comprising the steps of:

randomly selecting a first one of said plurality of home nodes;

fragmenting, by said selected home node, a query from a user into a plurality of query fragments;

hashing, by said selected home node, each said query fragment of said plurality of query fragments, said hashed query fragment having a first portion and a second portion;

transmitting, by said selected home node, each said hashed query fragment of said plurality of query fragments to a respective one of said plurality of query nodes indicated by said first portion of each said hashed query fragment;

using, by said query node, said second portion of said respective hashed query fragment to access data according to a local hash table located on said query node; and

returning, by each said query node accessing data according to said respective hashed query fragment, an object identifier corresponding to said accessed data to said selected home node.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL: (617) 542-2290
FAX: (617) 451-0313

6. (Amended) A method of storing data in a manner which is conducive to information retrieval using fuzzy queries in a distributed database system having a plurality of home nodes [node] and a plurality of query nodes connected by a network, said method comprising the steps of:

randomly selecting a first one of said plurality of home nodes;

fragmenting, by said selected home node, data from a user into a plurality of data fragments;

hashing, by said selected home node, each said data fragment of said plurality of data fragments, said hashed data fragment having a first portion and a second portion;

transmitting, by said selected home node, each said hashed data fragment of said plurality of data fragments to a respective one of said plurality of query nodes indicated by said first portion of each said hashed data fragment; and

using, by said query node, said second portion of said respective hashed data fragment to store data according to a local hash table located on said query node.

8. (Amended) A distributed database system having an information retrieval tool for handling queries from a user, comprising:

a plurality of home nodes [node]; and

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL: (617) 542-2290
FAX: (617) 451-0313

a plurality of query nodes;

said plurality of home nodes [node] and said plurality of query nodes connected by a network,

wherein each said home node, upon receiving a query from a user, fragments said query into a plurality of query fragments, hashes each said query fragment of said plurality of query fragments into a hashed query fragment having a first portion and a second portion, and transmits each said hashed query fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed query fragment, and

further wherein each said query node [,] uses said second portion of said hashed query fragment to access data according to a local hash table located on said query node and returns [,] an object identifier corresponding to said accessed data to said home node.

12. (Amended) A distributed database system for storage and retrieval of information, comprising:

a plurality of home nodes [node]; and

a plurality of query nodes;

said plurality of home nodes [node] and said plurality of query nodes connected by a network,

wherein each said home node, upon receiving data from a user, fragments said data into a plurality of data fragments, hashes each

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

said data fragment of said plurality of data fragments into a hashed data fragment having a first portion and a second portion, and transmits each said hashed data fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed data fragment, and

wherein each said query node [,] uses said second portion of said hashed data fragment to store data according to a local hash table located on said query node.

84

13. (Amended) A distributed database system having an information retrieval tool for handling queries from a user, comprising:

a plurality of home [node] nodes; and

a plurality of query nodes, said plurality of home nodes and said plurality of query nodes connected by a network,

each said home node, upon receiving a command from a user, enqueueing a predetermined task in response to said command,

a query task enqueued being resultant in, in response to a query command from said user, fragmenting a query contained in said query command into a plurality of query fragments, hashing each said query fragment of said plurality of query fragments into a hashed query fragment having a first portion and a second portion, and transmitting a query message containing each said hashed query fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed query fragment,

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

said query node, upon receipt of said query message, using said second portion of said hashed query fragment to access data according to a local hash table located on said query node and transmitting a message returning an object identifier corresponding to said accessed data to said home node.

17. (Amended) A distributed database system for storage and retrieval of information, comprising:

a plurality of home node nodes; and

a plurality of query nodes, said plurality of home nodes and said plurality of query nodes connected by a network,

each said home node, upon receiving a command from a user, enqueuing a predetermined task in response to said command,

an insert task enqueued, in response to an insert command from said user, fragmenting data contained in said insert command into a plurality of data fragments, hashing each said data fragment of said plurality of data fragments into a hashed data fragment having a first portion and a second portion, and transmitting an insert message containing each said hashed data fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed data fragment,

said query node, upon receipt of said insert message, using said second portion of said hashed data fragment to store data according to a local hash table located on said query node.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

R E M A R K S

The above-identified patent application has been amended and reconsideration is respectfully requested. Claims 1-17 are pending and stand rejected. Claims 1, 6, 8, 12, 13 and 17 have been amended.

Claims 1-17 are rejected under 35 U.S.C. §103 as being unpatentable over Chaturvedi, et al., "Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach", IEEE Transactions on Engineering Management, Vol. 41, No. 2, May 1994 and Houtsma et al., "Parallel Hierarchical Evaluation of Transitive Closure Queries", IEEE, 1991. With respect to claims 1, 6, 8, 12-13, and 17, the Examiner states that, "It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the hashing means of Houtsma's teachings with the teachings of Chaturvedi because the hashing means could enable Chaturvedi's information retrieval means to provide the queried node with a query value and a query identifier during the query nodes hashing process" (Paper No. 3, page 3). However, such a combination would not provide the distributed database and method of the present invention.

Both the Chaturvedi and Houtsma references describe techniques for partitioning files in a Distributed Relational Database System. These two references, and each of the papers cited by these two references, are in the field of relational database systems. A

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

relational database system consists of one or more relations, also known as tables or files. Each relation is a set of records, also known as rows or tuples. Each record in a relation has a set of attributes, also known as fields or columns. Every record in a relation has exactly the same number of fields and the fields have the same types. For example, a customer relation might consist of a 40 character name field, a 60 character address field and a 6 digit customer identifier.

A fundamental characteristic of relational databases is that records do **not have object identity**. More particularly, each record is uniquely determined by the values of its fields. By contrast, data models other than the relational model generally assume that the basic objects do have object identity, i.e., an object exists independently of any attribute values it might have, and changing the attribute values will not change the object identity.

Another fundamental characteristic of relational databases is the use of a relational query language called the relational algebra. The relational algebra is roughly equivalent to what mathematicians call the "first order predicate calculus," and is primarily used for extracting information from a relational database system. However, the relational algebra may also be used for other purposes. For example, relational algebra expressions can be used to specify database views, security and authentication

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

conditions, integrity constraints and database partitions. This can be confusing, and has apparently caused confusion in the examination of the present application, since these other uses of the relational algebra have nothing to do with extracting information from the database, and yet the word "query" is frequently used in connection with these other uses.

Modern relational databases typically deal with very large relations, i.e., relations that contain several terabytes (million megabytes) of data are common. The need to deal with such large relations along with the reduction in cost of computing equipment has driven the development of distributed relational database systems. A distributed relational database system is a relational database system that is distributed among a collection of computers which are connected by a communication network. Very large relations are distributed among the computers in the network by partitioning or otherwise breaking up the relations into disjoint pieces known as "fragments." These fragments are themselves relations, and typically contain in excess of tens or even hundreds of megabytes, even though the fragments are much smaller than the larger relation of which they are parts. Significantly, these relational fragments are **disjoint**.

The fragments of a distributed relational database system are defined by using the relational algebra. Perhaps as a result, the term "fragment query" is often used to refer to the relational

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

algebra expression that defines a relational database fragment. This can be confusing, and has apparently caused confusion in the examination of the present application, since the relational algebra expression "fragment query" does not describe extraction of information, but rather provides the defining condition for the fragment.

The present invention does not utilize the relational model, and in particular does not utilize the relational models of Chaturvedi and Houtsma. A primary purpose of the present invention is to allow information retrieval for information objects that are more general than the simple records of a relational model system. For example, documents such as papers, books, World Wide Web pages, annotated images, and other documents can all be indexed using a search engine in accordance with the present invention. Significantly, none of these documents would be considered searchable records according to the relational model.

The present invention and the relational model express queries and records differently. The query language used by the present invention is the same language used to express the information objects, or more precisely their content labels, that are indexed by the search engine of the present invention (claims 1, 6, 8, 12, 13 and 17). This has the advantage that no additional language is required for expressing queries. In contrast, relational database system queries are expressed in the relational algebra and the

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

records are expressed in other ways. The result of a query provided to the search engine of the present invention is a set of object identifiers (claim 1, line 17) with weights (claim 3, lines 2-3, claim 9) attached thereto. The weight attached to an object identifier represents ambiguous and fragmentary queries, which are also known as "fuzzy" queries. There is no analogous concept in the relational algebra. A relational algebra expression is a precise and unambiguous specification of a set of records. Using colloquial language, there is no "fuzziness" in the relational algebra.

The fragmentation technique of the present invention is different from fragmentation in the relational model. The present invention introduces a fragmentation technique that is utilized in the indexing algorithm. Information objects, or more precisely their content labels, are broken up into a collection of small overlapping fragments (claim 1, lines 4-5). The size of each fragment may typically be around 20 bytes. By contrast, the fragments of the relational model never overlap, are millions of times larger, and have a structure that is both conceptually and practically different. Furthermore, the present invention fragments both queries and information objects in the same way. This is impossible for relational model database systems, since queries and records have different structures.

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL. (617) 542-2290
FAX (617) 451-0313

With regard to the comment on Page 2, heading 3, sentence 1, Charturvedi introduces a new algorithm for defining fragments in a partitioned, distributed relational database system. As noted above, these relational fragments are unrelated to the object fragments of the present invention. This difference is illustrated by the cited example which uses the value of an attribute (named c) to break apart a base table (named T1) into two relation fragments, according to whether the attribute has value 'A' or 'B'.

With regard to the comment on Page 2, heading 3, sentence 2, Chaturvedi introduces a variation on the well known semijoin algorithm for computing a join. The join is one of the operators of the relational algebra, and computing it efficiently is important in relational database systems. Significantly, the algorithm for the two-way join described in Chaturvedi is very different from the algorithm used by the present invention. The Chaturvedi join query is split into two single-table sub-queries and then provided to the two nodes containing the base tables specified in the sub-queries. This splitting technique is commonly employed in Distributed Relational Database Systems. It is an algebraic factoring of the relational algebra expression that is the query. Algebraic factoring is a technique unrelated to the fragmentation of the present invention. More particularly, in the present invention each fragment is hashed in its entirety (claim 1, lines 6-8), and the hash value is provided to a node determined by

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

the hash value itself. In the splitting technique in Chaturvedi, sub-queries are not hashed at all; they are shipped to the node containing the base table specified in the sub-query. This is hardly surprising as it would not make any sense to hash a relational query because the resulting hash value would not have any uses.

With regard to the comments associated with Figs. 2 and 3 of Chaturvedi, the architecture of Chaturvedi shown in those Figs. is quite different from the architecture of the present invention. More particularly, there is no central server in the present invention, and neither the nodes of the network nor the object fragments in the index have any kind of hierarchical structure. In the present invention the home node of a query is randomly chosen, and different queries will generally have different home nodes.

With regard to the comment on Page 2, heading 3, sentence 3, the database fragmentation mentioned by Chaturvedi in the Abstract is relational fragmentation and is unrelated to the fragmentation of the present invention. The fragment queries in Chaturvedi's Illustrative Examples (Page 198) are not query fragments, but rather relational algebra expressions used to define relation fragments. Numbers 1-4 in Example 2 on page 198 are queries that are in the query history at Site A. They are queries that at some time in the past were processed at Site A. They are used by the MLTIF to compute relational algebra expressions for defining

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

relation fragments that would be better suited for evaluating the queries in the history than the current relation fragments. The presumption is that the past history is a good indicator of what the future will be. The MLTIF is not a query evaluation algorithm but rather a dynamic method for choosing good relation fragments in a Distributed Relational Database System. Therefore, the cited passages of the Chaturvedi reference are irrelevant to the Present invention.

With regard to the comment on Page 3, heading 3, sentence 1, nowhere on Page 197, column 1 or Page 199, column 2 of Chaturvedi is there any mention of a local hash table or any hashing operation.

With regard to the comment on Page 3, heading 3, sentence 2, no object identifiers are mentioned on Page 198 of Chaturvedi. Indeed, since the relational model explicitly rejects object identity, it would be amazing if it did mention object identifiers. The Illustrative Example on page 198 simply discusses how to find relational algebra expressions for defining time invariant relational fragments.

With regard to the comment on Page 3, heading 3, sentence 3, no hashing operation is mentioned anywhere in the Abstract.

With regard to the comment on Page 3, heading 3, sentence 4, Houtsma does not teach use of hashing. Indeed, on page 130, column 2, par. 3, Houtsma refers to a number of papers that use different

WITHROW, SCHURGIN, GAGHEIMER & HAYES
TEL. (617) 542-7700
FAX (617) 451-0313

methods to solve the transitive closure problem, including hash-based methods. Houtsma teaches a disconnection set approach that does not use hashing. Further, the graph shown in Houtsma is an auxiliary structure used in the algorithm. The graph defines a notion of adjacency between relation fragments. This is unrelated to the graphs (semantic networks) used in the present invention. As discussed above, the fragments of the present invention are quite different from the fragments of the relational model. Since the fragments of the present invention are parts of the semantic network, there is no concept of fragment adjacency in the present invention. In Houtsma, the graph has the relation fragments as the vertices, with unlabeled edges defined by relation fragment adjacency, while in the present invention the fragments may be regarded as fragments of a graph having labeled edges (semantic relationships) that connect concept instantiations with one another.

With regard to the comment on Page 3, heading 3, sentence 5, no hashing operation is mentioned here or anywhere in the reference. The fragment H is the high speed fragment. The term "high speed" was probably chosen because of their motivating example: the railway network of many European countries. It could equally well have been called the "special fragment" or the "wide-connection fragment."

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

The Examiner has also rejected claims 2, 7 and 14 based on the Chaturvedi and Houtsma references. However, Houtsma does not use hashing and Chaturvedi does not solve the information retrieval problem of the present invention. The Chaturvedi network architecture is very different from the architecture of the present invention. In Chaturvedi, except for the central server node, it is presumed that the servers are located where the queries will be presented by users. By contrast, the architecture of the present invention is a search engine that is entirely remote from any user nodes. The "home node" in Chaturvedi is the user node itself, i.e., the node where the query is presented to the distributed system. The "home node" in the present invention is one of the nodes in the search engine, and it can be randomly chosen by one of the front end processors. Further, Chaturvedi never fragments a query.

In addition to the architectural differences, there are no concepts of measure of relevance or degree of relevance (claims 3, 9) in the relational model, and no such concepts are mentioned or employed in Chaturvedi. In particular, the use of the word "relevance" in Chaturvedi is unrelated to the "fuzzy" notion of relevance in the present invention. Like all research on relational systems, Chaturvedi employs no notion of weighted relevance. When it is stated, for example, that "...it [join-value set] is transmitted to the relevant nodes participating in the join

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 542 2290
FAX (617) 451-0313

operation," Chaturvedi simply means that the join-value set is sent to those nodes participating in the join which may contribute any tuples to the result of the join. There is no relevance weighting involved in this operation. If it can be determined that a node participating in the join will not contribute any tuples to the result, then it is not sent the join-value set, otherwise the join-value set is sent to the node. The decision is completely "sharp" and does not involve any "fuzziness." This is hardly surprising since Chaturvedi describes a relational model which is unrelated to information retrieval using fuzzy queries.

With regard to fragment storage, the storage of relation fragments in a Distributed Relational Database System is specified in the allocation schema. In Chaturvedi, Example 4, there are three relation fragments: T1A, T1B, and T2. T1A is the relation fragment defined by the relational algebra expression:

SELECT * FROM T1 WHERE e = 'A'

and T1B is the relation fragment defined by the relational algebra expression:

SELECT * FROM T1 WHERE e = 'B'

The allocation schema simply specifies which nodes contain a copy of each relation fragment. Here, for example, is the allocation schema used by Chaturvedi in this example:

T1A: node A

T1B: node B

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

T2: node A

It is merely coincidence that the value of the attribute e coincides with the name of the node.

With regard to the comments on Page 4, the steps in Chaturvedi, page 199, column 1 are concerned with choosing time invariant relation queries. These steps are not concerned with query processing per se. In the present invention a query request can specify one of several levels of service (claim 16). Roughly speaking, the lower levels of service are faster but are less accurate, the higher levels of service are slower but more accurate. This notion of level of service is meaningless for the relational model. In the relational model, all queries have exactly one correct answer. There is no concept in the relational model of answers that are better or worse.

In sum, the field of "information retrieval using fuzzy queries" (a term of art) is quite different from the relational model. In the relational model a query is a complete and unambiguous specification of the result. Relevance in the relational model is either TRUE or FALSE. In information retrieval results are returned which may or may not satisfy the intentions behind the query, and which may even be unrelated to the intentions behind the query. The claims have been amended to particularly point out this difference and remove the confusion which has

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
TEL (617) 542-2290
FAX (617) 451-0313

apparently been brought about by the use of terms which are similar to those of the cited references.

For the reasons given above, reconsideration and allowance is respectfully requested. The Examiner is encouraged to telephone the undersigned attorney to discuss any matters in furtherance of the prosecution of this application.

Respectfully submitted,

Kenneth P. Baclawski

By _____
Stanley M. Schurgin
Registration No. 20,979
Attorney for Applicant(s)

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES
Ten Post Office Square
Boston, Massachusetts 02109

Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: 6/7/96

SMS/jet
84277

03/94 FORM 9

Serial No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2307

#4

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
Ten Post Office Square
Boston, Massachusetts 02109
Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: June 7, 1996

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Attorney
Docket No.: NU-360XX

Sir:

In re application of: Kenneth P. Baclawski

Entitled: *DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD*

Transmitted herewith is an **amendment** in the above-identified application. The following checked items are applicable:

[ ] A Petition for Extension of Time for ____ month(s) is hereby made, under §1.136(a); a check in the amount of $_____ is enclosed per §1.17.

[X] In the event a Petition for Extension of Time is required by this paper and not otherwise provided, such Petition is hereby made and authorization is provided herewith to charge Deposit Account No. 23-0804 for the cost of such extension.

[ ] _____ is hereby appointed Associate Attorney by:
Registration No.:

Attorney of Record:
Registration No.:

[ ] Other:

CLAIMS AFTER AMENDMENT:	MINUS PRIOR PAID CLAIMS:	EQUALS PRESENT EXTRA CLAIMS:	RATE:	ADDITIONAL FEE:
Independent	6 - 6	= 0	x $78.00 =	00
Total	11 - 11	= 0	x $22.00 =	0
[ ] Multiple Dependent claims (1st presentation)			+ $250.00 =	0
SUBTOTAL ADDITIONAL FEE				0
Small Entity filing, divide by 2. Previously submitted or verified statement must be attached (per §1.9, §1.27, §1.28)				0
TOTAL ADDITIONAL FEE				00

[X] *No additional fee.* [ ] The fee has been calculated above; a check in the amount of $.00 is enclosed.

[X] The Commissioner is hereby authorized to charge payment of any additional filing fees under §1.16 associated with this communication or credit any overpayment to Deposit Account No. 23-0804.

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on 6/7/96.

SUBMIT IN TRIPLICATE
85145

Attorney of Record: Stanley M. Schurgin
Registration No.: 20,979

03/94 FORM 9

Serial No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2307

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES
Ten Post Office Square
Boston, Massachusetts 02109
Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: June 7, 1996

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Attorney
Docket No.: NU-360XX

Sir:

In re application of: Kenneth P. Baclawski

Entitled: DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

Transmitted herewith is an **amendment** in the above-identified application. The following checked items are applicable:

[ ] A petition for Extension of Time for ____ month(s) is hereby made, under §1.136(a); a check in the amount of $_____ is enclosed per §1.17.

[X] In the event a Petition for Extension of Time is required by this paper and not otherwise provided, such Petition is hereby made and authorization is provided herewith to charge Deposit Account No. 23-0804 for the cost of such extension.

[ ] _____ is hereby appointed Associate Attorney by:
Registration No.:

_____
Attorney of Record:
Registration No.:

JUN 24 1996

[ ] Other:

CLAIMS AFTER AMENDMENT:	MINUS PRIOR PAID CLAIMS:	EQUALS PRESENT EXTRA CLAIMS:	RATE:	ADDITIONAL FEE:
Independent	6 - 6	= 0	x $78.00 =	00
Total	11 - 11	= 0	x $22.00 =	0
[ ] Multiple Dependent Claims (1st presentation)			+ $250.00 =	0
		SUBTOTAL ADDITIONAL FEE		0
Small Entity filing, divide by 2. Previously submitted or verified statement must be attached (per §1.9, §1.27, §1.28)				0
		TOTAL ADDITIONAL FEE		00

[X] No additional fee. [ ] The fee has been calculated above; a check in the amount of $.00 is enclosed.

[X] The Commissioner is hereby authorized to charge payment of any additional filing fees under §1.16 associated with this communication or credit any overpayment to Deposit Account No. 23-0804.

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on 6/7/96.

SUBMIT IN TRIPLICATE
35145

_____
Attorney of Record: Stanley M. Schurgin
Registration No.: 20,979

03/94 FORM 9

Ser. No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2307

..., SCHURGIN, GAGNEBIN & HAYES
...en Post Office Square
Boston, Massachusetts 02109
Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: June 7, 1996

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Attorney
Docket No.: HU-3601X

Sir:

In re application of: Kenneth P. Baclawski

Entitled: DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

Transmitted herewith is an amendment in the above-identified application. The following checked items are applicable:

[ ] A Petition for Extension of Time for ___ month(s) is hereby made, under §1.136(a); a check in the amount of $_____ is enclosed per §1.17.

[X] In the event a Petition for Extension of Time is required by this paper and not otherwise provided, such Petition is hereby made and authorization is provided herewith to charge Deposit Account No. 23-0804 for the cost of such extension.

[ ] _____ is hereby appointed Associate Attorney by:
Registration No.:

Attorney of Record:
Registration No.:

[ ] Other:

CLAIMS AFTER AMENDMENT:	MINUS PRIOR PAID CLAIMS:	EQUALS PRESENT EXTRA CLAIMS:	RATE:	ADDITIONAL FEE:
Independent	6 - 6	= 0	x $78.00 =	00
Total	11 - 11	= 0	x $22.00 =	0
[ ] Multiple Dependent Claims (1st presentation)			+ $250.00 =	0
		SUBTOTAL ADDITIONAL FEE		0
Small Entity filing, divide by 2. Previously submitted or verified statement must be attached (per §1.9, §1.27, §1.28)				0
		TOTAL ADDITIONAL FEE		00

[X] No additional fee. [ ] The fee has been calculated above; a check in the amount of $.00 is enclosed.

[X] The Commissioner is hereby authorized to charge payment of any additional filing fees under §1.16 associated with this communication or credit any overpayment to Deposit Account No. 23-0804.

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on 6/7/96.

SUBMIT IN TRIPLICATE
85145

Attorney of Record: Stanley M. Schurgin
Registration No.: 20,979

UB

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

SERIAL NUMBER	FILING DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NO.
08/318,252	10/05/94	BACI AWSKI K	NI360XX

EJM1/0911
WEINGARTEN SCHURGIN GAGNEBIN & HAYES
TEN POST OFFICE SQUARE
BOSTON MA 02109

LEWIS, EXAMINER	
ART UNIT	PAPER NUMBER
2307	5

DATE MAILED: 09/11/96

Please find below a communication from the EXAMINER in charge of this application.

Commissioner of Patents

See attached

PTOL-90 (Rev. 8/94)

1 - PATENT APPLICATION FILE COPY

Office Action Summary

Application No.	Applicant(s)
08/38,259	Bae [illegible]
Examiner	**Group Art Unit**
Cheryl Lewis	3307

☒ Responsive to communication(s) filed on [illegible]

☐ This action is FINAL.

☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11; 453 O.G. 213.

A shortened statutory period for response to this action is set to expire 3 month(s), or thirty days, whichever is longer, from the mailing date of this communication. Failure to respond within the period for response will cause the application to become abandoned. (35 U.S.C. § 133). Extensions of time may be obtained under the provisions of 37 CFR 1.136(a).

Disposition of Claims

☒ Claim(s) 1-17 is/are pending in the application.

Of the above, claim(s) _____ is/are withdrawn from consideration.

☐ Claim(s) _____ is/are allowed.

☒ Claim(s) 1-17 is/are rejected.

☐ Claim(s) _____ is/are objected to.

☐ Claims _____ are subject to restriction or election requirement.

Application Papers

☐ See the attached Notice of Draftsperson's Patent Drawing Review, PTO-948.

☐ The drawing(s) filed on _____ is/are objected to by the Examiner.

☐ The proposed drawing correction, filed on _____ is ☐ approved ☐ disapproved.

☐ The specification is objected to by the Examiner.

☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. § 119

☐ Acknowledgement is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d).

☐ All ☐ Some* ☐ None of the CERTIFIED copies of the priority documents have been

☐ received.

☐ received in Application No. (Series Code/Serial Number) _____

☐ received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

*Certified copies not received: _____

☐ Acknowledgement is made of a claim for domestic priority under 35 U.S.C. § 119(e).

Attachment(s)

☒ Notice of References Cited, PTO-892

☐ Information Disclosure Statement(s), PTO-1449, Paper No(s). _____

☐ Interview Summary, PTO-413

☐ Notice of Draftsperson's Patent Drawing Review, PTO-948

☐ Notice of Informal Patent Application, PTO-152

— SEE OFFICE ACTION ON THE FOLLOWING PAGES —

1. Claims 1-17 are presented for examination.

2. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.

3. Claims 1-17 are rejected under 35 U.S.C. 102(b) as being anticipated by Neches, Patent Number: 5,006,978.

4. - With respect to claims 1, 6-8, 12-14, and 17,

Neches discloses the random selection means for a plurality of home nodes (fig. 1), the home node fragmenting means including a query fragment (figs. 2-2a), the hashing means of the selected home node (figs. 2b-2c), and the query fragments local hash table means (figs. 2-2a) and returning means.

5. - With respect to claim 2,

Neches discloses the means to receive a home node including the query fragment means (figs. 2c-2d).

6. - With respect to claims 3 and 9,

Neches disclosea the means to determine a measure of relevance between the accessed data and query (figs. 2h-2j) and the returning step of the object identifier (fig. 2h.

7. - With respect to claims 4 and 10,

Neches disclosea the means which essentially comprise the same means as determining a measure or relevance by a cosine measure (fig. 5).

8. - With respect to claims 5 and 11,

Neches discloses the portion hashed query fragment to comprise 5 bits and 32 bits (fig. 11).

9. - With respect to claims 15 and 16,

Neches discloses the means to label and return three query levels (figs. 2-2j).

10. Applicant's arguments with respect to claims 1-17 have been considered but are moot in view of the new grounds of rejection.

11. Any inquiry concerning this communication or earlier communications form the examiner should be directed to Cheryl Lewis whose telephone number is (703) 305-8750.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-9600.

THOMAS C. BLACK
SUPERVISORY PATENT EXAMINER
GROUP 2300

Notice of References Cited

Application No.	Applicant(s)
08/38,252	Barlow et al.
Examiner	**Group Art Unit**
Cheryl Lewis	2307

Page 1 of 1

U.S. PATENT DOCUMENTS

*		DOCUMENT NO.	DATE	NAME	CLASS	SUBCLASS
	A	5,355,364	05/03/94	Katz et al.	370	389.09
	B	5,606,978	04/04/01	Nelms	364	240
	C					
	D					
	E					
	F					
	G					
	H					
	I					
	J					
	K					
	L					
	M					

FOREIGN PATENT DOCUMENTS

*		DOCUMENT NO.	DATE	COUNTRY	NAME	CLASS	SUBCLASS
	N						
	O						
	P						
	Q						
	R						
	S						
	T						

NON-PATENT DOCUMENTS

*		DOCUMENT (Including Author, Title, Source, and Pertinent Pages)	DATE
	U		
	V		
	W		
	X		

* A copy of this reference is not being furnished with this Office action.
(See Manual of Patent Examining Procedure, Section 707.05(a).)

#6/B
W. Lawson
1-7-97

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application	:	Kenneth P. Baclawski
Application No.	:	08/318,252
Filed	:	October 5, 1994
For	:	DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD
Examiner	:	C. Lewis
Attorney's Docket	:	NU-360XX

Group Art Unit: 2307

* *

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: BOX NON-FEE AMENDMENT, Assistant Commissioner for Patents, Washington, D.C. 20231 on 12/6/96.

By _____
Stanley M. Schurgin
Registration No. 20,979
Attorney for Applicant

* *

AMENDMENT

BOX NON-FEE AMENDMENT
Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

In response to the Office Action dated September 11, 1996, please amend the above-identified patent application as follows:

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

In the Claims

Please amend claims 1, 6, 7, 8, 12, 13 and 17 as follows:

1. (Twice Amended) A method for information retrieval using fuzzy queries in a non-relational, distributed database system having a plurality of home nodes and a plurality of query nodes connected by a network, said method comprising the steps of:

randomly selecting a first one of said plurality of home nodes;

fragmenting, by said selected home node, a query from a user into a plurality of query fragments;

hashing, by said selected home node, each said query fragment of said plurality of query fragments, said hashed query fragment having a first portion and a second portion;

transmitting, by said selected home node, each said hashed query fragment of said plurality of query fragments to a respective one of said plurality of query nodes indicated by said first portion of each said hashed query fragment;

using, by said query node, said second portion of said respective hashed query fragment to access data according to a local hash table located on said query node; and

returning, by each said query node accessing data according to said respective hashed query fragment, an object identifier corresponding to said accessed data to said selected home node.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

6. (Twice Amended) A method of storing [data] objects in a manner which is conducive to information retrieval using fuzzy queries in a non-relational, distributed database system having a plurality of home nodes and a plurality of query nodes connected by a network, said method comprising the steps of:

randomly selecting a first one of said plurality of home nodes;

fragmenting, by said selected home node, [data] objects from a user into a plurality of [data] object fragments;

hashing, by said selected home node, each said [data] object fragment of said plurality of [data] object fragments, said hashed [data] object fragment having a first portion and a second portion;

transmitting, by said selected home node, each said hashed [data] object fragment of said plurality of data fragments to a respective one of said plurality of query nodes indicated by said first portion of each said hashed [data] object fragment; and

using, by said query node, said second portion of said respective hashed [data] object fragment to store data according to a local hash table located on said query node.

7. (Amended) The method of claim 6 further comprising the step of receiving, at said home node, said [data] objects from said user, prior to the step of fragmenting said [data] object.

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL. (617) 542-2290
FAX (617) 451-0313

8. (Twice Amended) A non-relational, distributed database system having an information retrieval tool for handling queries from a user, comprising:

a plurality of home nodes; and

a plurality of query nodes;

said plurality of home nodes and said plurality of query nodes connected by a network,

wherein each said home node, upon receiving a query from a user, fragments said query into a plurality of query fragments, hashes each said query fragment of said plurality of query fragments into a hashed query fragment having a first portion and a second portion, and transmits each said hashed query fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed query fragment, and

further wherein each said query node uses said second portion of said hashed query fragment to access data according to a local hash table located on said query node and returns an object identifier corresponding to said accessed data to said home node.

12. (Twice Amended) A non-relational, distributed database system for storage and retrieval of information objects, comprising:

a plurality of home nodes; and

a plurality of query nodes;

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

said plurality of home nodes and said plurality of query nodes connected by a network,

wherein each said home node, upon receiving [data] an object from a user, fragments said [data] object into a plurality of [data] object fragments, hashes each said [data] object fragment of said plurality of [data] object fragments into a hashed [data] object fragment having a first portion and a second portion, and transmits each said hashed [data] object fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed [data] object fragment, and

wherein each said query node uses said second portion of said hashed [data] object fragment to store [data] objects according to a local hash table located on said query node.

13. (Twice Amended) A non-relational, distributed database system having an information retrieval tool for handling queries from a user, comprising:

a plurality of home nodes; and

a plurality of query nodes, said plurality of home nodes and said plurality of query nodes connected by a network,

each said home node, upon receiving a command from a user, enqueueing a predetermined task in response to said command,

a query task enqueued being resultant in, in response to a query command from said user, fragmenting a query contained in said

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

query command into a plurality of query fragments, hashing each said query fragment of said plurality of query fragments into a hashed query fragment having a first portion and a second portion, and transmitting a query message containing each said hashed query fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed query fragment,

said query node, upon receipt of said query message, using said second portion of said hashed query fragment to access data according to a local hash table located on said query node and transmitting a message returning an object identifier corresponding to said accessed data to said home node.

17. (Twice Amended) A non-relational, distributed database system for storage and retrieval of information, comprising:

a plurality of home node nodes; and

a plurality of query nodes, said plurality of home nodes and said plurality of query nodes connected by a network,

each said home node, upon receiving a command from a user, enqueueing a predetermined task in response to said command,

an insert task enqueued, in response to an insert command from said user, fragmenting data contained in said insert command into a plurality of data fragments, hashing each said data fragment of said plurality of data fragments into a hashed data fragment having a first portion and a second portion, and transmitting an insert

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
TEL. (617) 542-2290
FAX (617) 451-0313

message containing each said hashed data fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed data fragment,

said query node, upon receipt of said insert message, using said second portion of said hashed data fragment to store data according to a local hash table located on said query node.

REMARKS

Claims 1-17 are pending in this application. Claims 1, 6, 8, 12, 13 and 17 have been amended.

The Examiner has rejected claims 1-17 under 35 U.S.C. § 102(b) as being anticipated by Neches, U.S. Patent No. 5,006,978. Neches teaches breaking up and distributing a relational database with a hash function for facilitation of data storage. Claims 1-5, 8-11 and 13-17 of the present application do not relate to storage of data, and are therefore not suggested by Neches. Claims 6, 7 and 12 of the present application relate to storage of data. However, claims 6, 7 and 12 are distinguished from Neches since these claims (as amended) recite method and apparatus for fragmenting, hashing and distributing objects. Operating upon objects is significantly different from operating upon relations because of size, content and structural differences. For example, a relation will typically be much larger than an object, and will not include methods.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

Relational database systems consist of relations, sometimes referred to as tables or files, where each such relations is a set of records, sometimes referred to as rows or tuples. Each record in such a relation has a set of attributes, also known as fields or columns. Significantly, each record in a relation has exactly the same number of fields, and the fields have the same types. For example, a customer relation could consist of a forty character name field, a sixty character address field and six digit customer identifier. Further, records in relational database systems do not have object identity. More particularly, each record is uniquely determined by the values of its fields.

The present invention expresses queries and records differently than the relational databases of Neches and the previously cited references. The query language used by the present invention is used to express information objects that are indexed by the search engine. In contrast, relational database queries are expressed in a relational algebra and records are expressed in other ways. The present invention therefore provides an advantage since separate "languages" are not required for expressing queries and records. Further, the result of a query provided to the search engine of the present invention is a set of object identifiers with weights attached thereto. Such results do not necessarily contain each term in the query or provide relevant information, and are therefore known as "fuzzy" queries. In

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL. (617) 542-2290
FAX (617) 451-0313

contrast, relational algebra expressions return a precise and unambiguous set of records, each of which is relevant since it satisfies each term in the relational algebra, and hence there is no "fuzziness" in the relational database model. The claims therefore recite these distinguishing features.

For the reasons stated above it is suggested that claims 1-17 are allowable, and reconsideration and allowance are respectfully requested. The Examiner is invited to telephone the undersigned attorney to discuss any matters which would expedite allowance of present application.

Respectfully submitted,

KENNETH P. BACIAWSKI

By _____
Stanley M. Schurgin
Registration No. 20,979
Attorney for Applicant

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
Ten Post Office Square
Boston, Massachusetts 02109

Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: 12/4/96

SMS/jet
93808

03/94 FORM 9

Se .l No.: 08/318,252
Fi .: October 5, 1994
Group Art Unit: 2307

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
Ten Post Office Square
Boston, Massachusetts 02109
Telephone: (617) 542-2290
Telecopier: (617) 451-0313

#6

Date: December 11, 1996

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Attorney
Docket No.: NU-360XX

Sir:

In re application of: Kenneth P. Baclawski

Entitled: DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

Transmitted herewith is an **amendment** in the above-identified application. The following checked items are applicable:

[ ] A Petition for Extension of Time for ___ month(s) is hereby made, under §1.136(a); a check in the amount of $_____ is enclosed per §1.17.

[X] In the event a Petition for Extension of Time is required by this paper and not otherwise provided, such Petition is hereby made and authorization is provided herewith to charge Deposit Account No. 23-0804 for the cost of such extension.

[ ] _____ is hereby appointed Associate Attorney by:
Registration No.:

RECEIVED

Attorney of Record:
Registration No.:

JAN 02 1997

GROUP 2300

[ ] Other:

CLAIMS AFTER AMENDMENT:	MINUS PRIOR PAID CLAIMS:	EQUALS PRESENT EXTRA CLAIMS:	RATE:	ADDITIONAL FEE:
Independent	6 - 6	= 0	x $80.00 =	00
Total	11 - 11	= 0	x $22.00 =	0
[ ] Multiple Dependent Claims (1st presentation)			+ $260.00 =	0
[redacted]				0
Small Entity filing, divide by 2. Previously submitted or verified statement must be attached (per §1.9, §1.27, §1.28)				0
[redacted]				00

[X] No additional fee. [ ] The fee has been calculated above; a check in the amount of $.00 is enclosed.

[X] The Commissioner is hereby authorized to charge payment of any additional filing fees under §1.16 associated with this communication or credit any overpayment to Deposit Account No. 23-0804.

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on 12/11/96.

SUBMIT IN TRIPLICATE
9a578

Attorney of Record: Stanley M. Schurgin
Registration No.: 20,979

03/94 FORM 9

Se 1 No.: 08/318,252
Fi_d: October 5, 1994
Group Art Unit: 2307

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
Ten Post Office Square
Boston, Massachusetts 02109
Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: December 11, 1996

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Attorney
Docket No.: NU-360XX

Sir:

In re application of: Kenneth P. Baclawski

Entitled: DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

Transmitted herewith is an **amendment** in the above-identified application. The following checked items are applicable:

[ ] A Petition for Extension of Time for ___ month(s) is hereby made, under §1.136(a); a check in the amount of $_____ is enclosed per §1.17.

[X] In the event a Petition for Extension of Time is required by this paper and not otherwise provided, such Petition is hereby made and authorization is provided herewith to charge Deposit Account No. 23-0804 for the cost of such extension.

[ ] _____ is hereby appointed Associate Attorney. RECEIVED
Registration No.:

JAN 02 1997

Attorney of Record:
Registration No.:

GROUP 2300

[ ] Other:

CLAIMS AFTER AMENDMENT:	MINUS PRIOR PAID CLAIMS:	EQUALS PRESENT EXTRA CLAIMS:	RATE:	ADDITIONAL FEE:
Independent	6 - 6	= 0	x $80.00 =	00
Total	11 - 11	= 0	x $22.00 =	0
[ ] Multiple Dependent Claims (1st presentation)			+ $260.00 =	0
			SUBTOTAL ADDITIONAL FEE	0
Small Entity filing, divide by 2. Previously submitted or verified statement must be attached (per §1.9, §1.27, §1.28)				0
			TOTAL ADDITIONAL FEE	00

[X] No additional fee. [ ] The fee has been calculated above; a check in the amount of $.00 is enclosed.

[X] The Commissioner is hereby authorized to charge payment of any additional filing fees under §1.16 associated with this communication or credit any overpayment to Deposit Account No. 23-0804.

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on 12/11/96.

SUBMIT IN TRIPLICATE
94578

Attorney of Record: Stanley M. Schurgin
Registration No.: 20,979

03/94 PO...

Se... l No.: 08/318,252
Fi...g: October 5, 1994
Group Art Unit: 2307

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
Ten Post Office Square
Boston, Massachusetts 02109
Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: December 11, 1996

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Attorney
Docket No.: NU-360XX

Sir:

In re application of: Kenneth P. Baclawski

Entitled: DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

Transmitted herewith is an **amendment** in the above-identified application. The following checked items are applicable:

[ ] A Petition for Extension of Time for ___ month(s) is hereby made, under §1.136(a); a check in the amount of $_____ is enclosed per §1.17.

[X] In the event a Petition for Extension of Time is required by this paper and not otherwise provided, such Petition is hereby made and authorization is provided herewith to charge Deposit Account No. 23-0804 for the cost of such extension.

[ ] _____ is hereby appointed Associate Attorney by:
Registration No.:

RECEIVED
JAN 0 2 1997
GROUP 2300

Attorney of Record:
Registration No.:

[ ] Other:

CLAIMS AFTER AMENDMENT:	MINUS PRIOR PAID CLAIMS:	EQUALS PRESENT EXTRA CLAIMS:	RATE:	ADDITIONAL FEE:
Independent	6 - 6	= 0	x $80.00 =	00
Total	11 - 11	= 0	x $22.00 =	0
[ ] Multiple Dependent Claims (1st presentation)			+ $260.00 =	0
		SUBTOTAL ADDITIONAL FEE		0
Small Entity filing, divide by 2. Previously submitted or verified statement must be attached (per §1.9, §1.27, §1.28)				0
		TOTAL AMOUNT ENCLOSED		00

[X] No additional fee. [ ] The fee has been calculated above; a check in the amount of $.00 is enclosed.

[X] The Commissioner is hereby authorized to charge payment of any additional filing fees under §1.16 associated with this communication or credit any overpayment to Deposit Account No. 23-0804.

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on 12/11/96.

SUBMIT IN TRIPLICATE
94578

Attorney of Record: Stanley M. Schurgin
Registration No.: 20,979

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.
08/318,252	10/05/94	DACLAWSKI K	NU3S01X

BGM1/0321
WEINGARTEN SCHURGIN GAGNEBIN & HAYES
TEN POST OFFICE SQUARE
BOSTON MA 02109

EXAMINER	
LEWIS,C	
ART UNIT	PAPER NUMBER
2307	

DATE MAILED: 03/21/97 #7

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner of Patents and Trademarks

PTO-90C (Rev. 2/93) ☆ U.S. GOVERNMENT PRINTING OFFICE 1996-411-876/40075 1- File Copy

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

SERIAL	FILING	FIRST NAME APPLICANT	ATTORNEY DOCKET
08/318252	10/05/94	BACLAWSKI K	NU360XX

WEINGARTEN, SCHORGIN, GAGNEBIN, & HAYES
TEN POST OFFICE SQUARE
BOSTON, MASSACHUSETTS 02109

EXAMINER	
LEWIS, C	
ART UNIT	PAPER
2307	7

DATE MAILED:

Please find below a communication from the Examiner in charge of this application.

Please, see attached documents.

Commissioner of Patents and Trademarks.

PTOL-90 (REV. 6/84)

Office Action Summary

Application No.	Applicant(s)
08/316,262	Baclawski
Examiner	**Group Art Unit**
Cheryl Lewis	2307

☒ Responsive to communication(s) filed on *Dec 12, 1996*

☐ This action is FINAL.

☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11; 453 O.G. 213.

A shortened statutory period for response to this action is set to expire *3* month(s), or thirty days, whichever is longer, from the mailing date of this communication. Failure to respond within the period for response will cause the application to become abandoned. (35 U.S.C. § 133). Extensions of time may be obtained under the provisions of 37 CFR 1.136(a).

Disposition of Claims

☒ Claim(s) *1-17* is/are pending in the application.

Of the above, claim(s) _____ is/are withdrawn from consideration.

☒ Claim(s) *3-5, 9-11, and 14-16* is/are allowed.

☒ Claim(s) *1, 2, 6-8, 12, 13, and 17* is/are rejected.

☐ Claim(s) _____ is/are objected to.

☐ Claims _____ are subject to restriction or election requirement.

Application Papers

☐ See the attached Notice of Draftsperson's Patent Drawing Review, PTO-948.

☐ The drawing(s) filed on _____ is/are objected to by the Examiner.

☐ The proposed drawing correction, filed on _____ is ☐ approved ☐ disapproved.

☐ The specification is objected to by the Examiner.

☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. § 119

☐ Acknowledgement is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d).

☐ All ☐ Some* ☐ None of the CERTIFIED copies of the priority documents have been

☐ received.

☐ received in Application No. (Series Code/Serial Number) _____.

☐ received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

*Certified copies not received: _____

☐ Acknowledgement is made of a claim for domestic priority under 35 U.S.C. § 119(e).

Attachment(s)

☒ Notice of References Cited, PTO-892

☐ Information Disclosure Statement(s), PTO-1449, Paper No(s). _____

☐ Interview Summary, PTO-413

☐ Notice of Draftsperson's Patent Drawing Review, PTO-948

☐ Notice of Informal Patent Application, PTO-152

— SEE OFFICE ACTION ON THE FOLLOWING PAGES —

Response to Amendment

1. This Office Action is in response to the applicant's communication filed December 12, 1998.

2. Claims 1-17 are presented for examination.

3. Applicant's have amended claims 1, 6, 8, 12, 13, and 17.

4. Claims 3-5, 9-11, and 14-16 are allowed over the prior art of record.

5. Applicant's arguments with respect to 1, 2, 6, 7, 8, 12, 13, and 17 claims have been considered but are deemed to be moot in view of the new grounds of rejection.

6. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office Action.

USC 102 Rejection

7. Claims 1, 2, 6, 7, 8, 12, 13,. and 17 are rejected under 35 USC 102(b) as being unpatentable over Kuechler et al., Patent Number: 4,811,199.

8. With respect to claims 1, 6, 7, 8, 12, 13, and 17,

Kuechler discloses randomly selecting home nodes and query nodes (col. 4, lines 18-34, col. 12, & lines 1-24). Kuechler discloses the means which essentially comprise the same means as hashing and fragmenting (col. 6, lines 24-31 & 51-68, col.7, & lines 1-14). Kuechler discloses the query fragments (fig. 1, item 32, col. 6 & lines 10-13, 24-31, 38-41), transmitting the query fragments (fig. 1, items 10-22), and the plurality of query nodes (col. 16 & lines 20/Salary Range Definitions/Salary Acceptable Array, 37/Job-Id Range Definition/Job-ID Acceptable Array, & 55/Name,

Salary, & Job-Id). Kuechler discloses the hashing table means (col. 13, lines 62-67, col. 14, & lines 1-17 & 47-55). Kuechler discloses returning the query node accessed to items respective hashed query fragment (Abstract, lines 8-15). Kuechler discloses the object identifier means corresponding to the accessed data (col. 8 & line 50, Name, Salary, & Job-Id).

9. - With respect to claim 2,

Kuechler discloses receiving at the home node a query from a user prior to the fragmenting step (col. 9 & lines 34-55).

Conclusion

11. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Cheryl Lewis whose telephone number is (703) 305-8750.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-9600.

CL
March 13, 1997

THOMAS A. DILLON
SUPERVISORY PATENT EXAMINER
GROUP 2300

Notice of References Cited

Application No.	Applicant(s)
08/318,252	Beclawski

Examiner	Group Art Unit	Page 1 of 1
Cheryl Lewis	2307	

U.S. PATENT DOCUMENTS

	DOCUMENT NO.	DATE	NAME	CLASS	SUBCLASS
A	4,811,193	03/07/89	Kuechler et al.	364	200
B					
C					
D					
E					
F					
G					
H					
I					
J					
K					
L					
M					

FOREIGN PATENT DOCUMENTS

	DOCUMENT NO.	DATE	COUNTRY	NAME	CLASS	SUBCLASS
N						
O						
P						
Q						
R						
S						
T						

NON-PATENT DOCUMENTS

	DOCUMENT (Including Author, Title, Source, and Pertinent Pages)	DATE
U		
V		
W		
X		

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application	:	Kenneth P. Baclawski
Application No.	:	08/318,252
Filed	:	October 5, 1994
For	:	DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD
Examiner	:	C. Lewis
Attorney's Docket	:	NU-360XX

Group Art Unit: 2307

* *

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: BOX NON-FEE AMENDMENT, Assistant Commissioner for Patents, Washington, D.C. 20231 on 5/19/97.

By _____
Stanley M. Schurgin
Registration No. 20,979
Attorney for Applicant

* *

AMENDMENT

BOX NON-FEE AMENDMENT
Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

In response to the Office Action dated March 21, 1997, reconsideration is respectfully requested in view of the following remarks:

REMARKS

Claims 1-17 are pending in this application. Applicant is pleased to acknowledge allowance of claims 3-5, 9-11 and 14-16. Claims 1, 2, 6-8, 12, 13 and 17 have been rejected in view of Kuechler. However, the present invention as claimed is patentably distinct from Kuechler.

As described at various points throughout columns 1-20, Kuechler employs a single node system for storing and manipulating information. At column 20, lines 60-68 and column 21, lines 1-30 Kuechler discusses a distributed version of the disclosed method. However, even in this distributed version Kuechler *only describes* employing the same node as the home node. Hence, Kuechler makes no distinction between a home node and a query node as recited in each of the independent claims of the present invention.

In addition to failing to distinguish home nodes from query nodes, Kuechler broadcasts the same query to every processing node (column 21, lines 9-10). Hence, the query is not fragmented as recited in the claims of the present invention. Further, the information elements (i.e., records) are distributed by storing whole records on the processing nodes, and these information elements are also not fragmented. The location of an information element is determined by its record number, not by any information contained in the record. By contrast, the present invention

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

describes a fundamentally distributed technique, and both queries and objects are fragmented. In the present invention query fragments are processed only on the node for which the query fragment is relevant, query fragments are not broadcast to all the nodes, objects are fragmented, and the information content of an object fragment is used to determine on which node it is to be stored. Further, objects are not stored on a single node. Because objects are fragmented and because these fragments are stored independently, objects are distributed over many nodes. These distinguishing features are recited in the claims and hence distinguish the present invention from Kuechler.

The Kuechler concept of a query is the one used by the relational model. Such a query is unambiguous in the sense that every record either satisfies the query or it does not. There is no "fuzziness." The Kuechler query processing technique does introduce additional records that may or may not satisfy the query, but this is done for the sake of improving performance, not because there is any fuzziness in the query. A final filtering step (Fig.1 item 32) removes the spurious records. By contrast, the present invention employs an intrinsically "fuzzy" notion of query. Objects satisfy the query to a greater or lesser degree. Higher levels of service in the present invention are designed to improve the estimates of the degrees by which objects satisfy the query

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

rather than to eliminate spurious objects. Such higher levels of service are optional, whereas the final filtering step of Kuechler is mandatory. Furthermore, the distribution of processing effort for the higher levels of service in the present invention are very different from the distribution of processing effort for the final filtering step in Kuechler. Kuechler assigns compact symbols or codes (Abstract, line 7 and column 8, lines 6-7) to ranges of attribute values. These codes are assigned unique codes. They are very different from hash values, which are computed, not assigned, and which are not unique. Finally, Kuechler does not use any hashing techniques. The topological maps of Kuechler are stored using some form of bit map (column 17, lines 51-61) rather than using a hash table.

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

For the reasons stated above it is submitted that claims 1, 2, 6-8, 12, 13 and 17 are allowable, and reconsideration and allowance are respectfully requested. The Examiner is invited to telephone the undersigned attorney to discuss any matters which would expedite allowance of present application.

Respectfully submitted,

KENNETH P. BACLAWSKI

By _____
Stanley M. Schurgin
Registration No. 20,979
Attorney for Applicant

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
Ten Post Office Square
Boston, Massachusetts 02109

Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: 5/09/97

SMS/rec
101555

9/96 FORM 9

Application No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2307

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
Ten Post Office Square
Boston, Massachusetts 02109
Telephone: (617) 542-2290
Telecopier: (617) 451-0313

#8

Date: May 14, 1997

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Attorney
Docket No.: NU-360XX

Sir:

In re application of: Kenneth P. Bazlawski

Entitled: DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

Transmitted herewith is an **amendment** in the above-identified application. The following checked items are applicable:

[ ] A Petition for Extension of Time for ___ month(s) is hereby made, under §1.136(a); a check in the amount of $_____ is enclosed per §1.17.

[X] In the event a Petition for Extension of Time is required by this paper and not otherwise provided, such Petition is hereby made and authorization is provided herewith to charge Deposit Account No. 23-0804 for the cost of such extension.

[ ] _____ is hereby appointed Associate Attorney by:
Registration No.:

Attorney of Record:
Registration No.:

[ ] Other:

CLAIMS AFTER AMENDMENT:	MINUS PRIOR PAID CLAIMS:	EQUALS PRESENT EXTRA CLAIMS:	RATE:	ADDITIONAL FEE:
Independent	6 - 6	0 = 0	x $50.00 =	0
Total	17 - 20	0 = 0	x $22.00 =	0
[ ] Multiple Dependent Claims (1st presentation)			+ $260.00 =	0
[redacted]				0
Small Entity filing, divide by 2. Previously submitted or verified statement must be attached (per §1.9, §1.27, §1.28)				0
[redacted]				0

[X] No additional fee. [ ] The fee has been calculated above; a check in the amount of $_____ is enclosed.

[X] The Commissioner is hereby authorized to charge payment of any additional filing fees under §1.16 associated with this communication or credit any overpayment to Deposit Account No. 23-0804.

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on 5/14/97.

Attorney of Record: Stanley N. Schurgin
Registration No.: 20,979

SUBMIT IN TRIPLICATE

102059

09/96 FORM 9

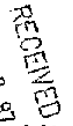
Application No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2307

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
Ten Post Office Square
Boston, Massachusetts 02109
Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: May 14, 1997

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Attorney
Docket No.: MU-350XX



Sir:

In re application of: Kenneth P. Baclawski

Entitled: DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

Transmitted herewith is an **amendment** in the above-identified application. The following checked items are applicable:

[ ] A Petition for Extension of Time for __ month(s) is hereby made, under §1.136(a); a check in the amount of $_____ is enclosed per §1.17.

[X] In the event a Petition for Extension of Time is required by this paper and not otherwise provided, such Petition is hereby made and authorization is provided herewith to charge Deposit Account No. 23-0804 for the cost of such extension.

[ ] _____ is hereby appointed Associate Attorney by:
Registration No.:

Attorney of Record:
Registration No.:

[ ] Other:

CLAIMS AFTER AMENDMENT:	MINUS PRIOR PAID CLAIMS:	EQUALS PRESENT EXTRA CLAIMS:	RATE:	ADDITIONAL FEE:
Independent	6 - 6	0 = 0	x $80.00 =	0
Total	17 - 20	0 = 0	x $22.00 =	0
[ ] Multiple Dependent Claims (1st presentation)			+ $260.00 =	0
		SUBTOTAL ADDITIONAL FEE		0
Small Entity filing, divide by 2. Previously submitted or verified statement must be attached (per §1.9, §1.27, §1.28)				0
		TOTAL ADDITIONAL FEE		0

[X] No additional fee. [ ] The fee has been calculated above; a check in the amount of $_____ is enclosed.

[X] The Commissioner is hereby authorized to charge payment of any additional filing fees under §1.16 associated with this communication or credit any overpayment to Deposit Account No. 23-0804.

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on 5/14/97.

SUBMIT IN TRIPLICATE

Attorney of Record: Stanley M. Schurgin
Registration No.: 20,979

102059

09/96 FORM 9

Application No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2307

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
Ten Post Office Square
Boston, Massachusetts 02109
Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: May 14, 1997

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Attorney Docket No.: NU-360XX

RECEIVED
JUN -2 97
GROUP 2600

Sir:

In re application of: Kenneth P. Baclawski

Entitled: DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

Transmitted herewith is an **Amendment** in the above-identified application. The following checked items are applicable:

[ ] A Petition for Extension of Time for ___ month(s) is hereby made, under §1.136(a); a check in the amount of $_____ is enclosed per §1.17.

[X] In the event a Petition for Extension of Time is required by this paper and not otherwise provided, such Petition is hereby made and authorization is provided herewith to charge Deposit Account No. 23-0804 for the cost of such extension.

[ ] _____ is hereby appointed Associate Attorney by:
Registration No.:

_____
Attorney of Record:
Registration No.:

[ ] Other:

CLAIMS AFTER AMENDMENT:	MINUS PRIOR PAID CLAIMS:	EQUALS PRESENT EXTRA CLAIMS:	RATE:	ADDITIONAL FEE:
Independent	6 - 6	0 = 0	x $80.00 =	0
Total	17 - 20	0 - 0	x $22.00 =	0
[ ] Multiple Dependent Claims (1st presentation)			+ $260.00 =	0
SUBTOTAL ADDITIONAL FEE				0
Small Entity filing, divide by 2. Previously submitted or verified statement must be attached (per §1.9, §1.27, §1.28)				0
TOTAL ADDITIONAL FEE				0

[X] No additional fee. [ ] The fee has been calculated above; a check in the amount of $_____ is enclosed.

[x] The Commissioner is hereby authorized to charge payment of any additional filing fees under §1.16 associated with this communication or credit any overpayment to Deposit Account No. 23-0804.

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on 5/14/97.

SUBMIT IN TRIPLICATE

Attorney of Record: Stanley M. Schurgin
Registration No.: 20,979

102059

Notice of Allowability	Application No. 08/318,252	Applicant(s) Beckwaki
	Examiner Cheryl Lewis	Group Art Unit 2307

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance and Issue Fee Due or other appropriate communication will be mailed in due course.

☒ This communication is responsive to *communication filed on May 19, 1997*.

☒ The allowed claim(s) is/are *1-17*.

☐ The drawings filed on _____ are acceptable.

☐ Acknowledgement is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d).

☐ All ☐ Some* ☐ None of the CERTIFIED copies of the priority documents have been

☐ received.

☐ received in Application No. (Series Code/Serial Number) _____.

☐ received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

*Certified copies not received: _____.

☐ Acknowledgement is made of a claim for domestic priority under 35 U.S.C. § 119(e).

A SHORTENED STATUTORY PERIOD FOR RESPONSE to comply with the requirements noted below is set to EXPIRE THREE MONTHS FROM THE "DATE MAILED" of this Office action. Failure to timely comply will result in ABANDONMENT of this application. Extensions of time may be obtained under the provisions of 37 CFR 1.136(a).

☐ Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL APPLICATION, PTO-152, which discloses that the oath or declaration is deficient. A SUBSTITUTE OATH OR DECLARATION IS REQUIRED.

☒ Applicant MUST submit NEW FORMAL DRAWINGS

☒ because the originally filed drawings were declared by applicant to be informal.

☐ including changes required by the Notice of Draftsperson's Patent Drawing Review, PTO-948, attached hereto or to Paper No. _____.

☐ including changes required by the proposed drawing correction filed on _____, which has been approved by the examiner.

☐ including changes required by the attached Examiner's Amendment/Comment.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the reverse side of the drawings. The drawings should be filed as a separate paper with a transmittal letter addressed to the Official Draftsperson.

☐ Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Any response to this letter should include, in the upper right hand corner, the APPLICATION NUMBER (SERIES CODE/SERIAL NUMBER). If applicant has received a Notice of Allowance and Issue Fee Due, the ISSUE BATCH NUMBER and DATE of the NOTICE OF ALLOWANCE should also be included.

Attachment(s)

☐ Notice of References Cited, PTO-892

☒ Information Disclosure Statement(s), PTO-1449, Paper No(s). 2

☐ Notice of Draftsperson's Patent Drawing Review, PTO-948

☐ Notice of Informal Patent Application, PTO-152

☐ Interview Summary, PTO-413

☐ Examiner's Amendment/Comment

☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material

☐ Examiner's Statement of Reasons for Allowance

THOMAS G. BLACK
SUPERVISORY PATENT EXAMINER
GROUP 2300

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office

Address: Box ISSUE FEE
ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

NOTICE OF ALLOWANCE AND ISSUE FEE DUE

24M1/0619
WEINGARTEN SCHURGIN GAGNEBIN & HAYES
TEN POST OFFICE SQUARE
BOSTON MA 02109

APPLICATION NO.	FILING DATE	TOTAL CLAIMS	EXAMINER AND GROUP ART UNIT		DATE MAILED
08/318,252	10/05/94	017	LEWIS, C	2307	06/19/97

First Named Applicant	BACLAWSKI,	KENNETH P.

TITLE OF INVENTION: DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

ATTY'S DOCKET NO.	CLASS/SUBCLASS	BATCH NO.	APPLN. TYPE	SMALL ENTITY	FEE DUE	DATE DUE
2 NU360XX	395-605.000	031	UTILITY	YES	$645.00	09/19/97

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED.

THE ISSUE FEE MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED.

HOW TO RESPOND TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.
If the SMALL ENTITY is shown as yes, verify your current SMALL ENTITY status:

A. If the status is changed, pay twice the amount of the FEE DUE shown and notify the Patent and Trademark Office of the change in status, or
B. If the status is the same, pay the FEE DUE shown above.

If the SMALL ENTITY is shown as NO:

A. Pay FEE DUE shown above, or

B. File verified statement of Small Entity Status before, or with, payment of 1/2 the FEE DUE shown above.

II. Part B of this notice should be completed and returned to the Patent and Trademark Office (PTO) with your ISSUE FEE. Even if the ISSUE FEE has already been paid by charge to deposit account, Part B should be completed and returned. If you are charging the ISSUE FEE to your deposit account, section "6b" of Part B should be completed.

III. All communications regarding this application must give application number and batch number. Please direct all communication prior to issuance to Box ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PTOL-85 (REV. 08-96) (0651-0033) 3. PATENT AND TRADEMARK OFFICE COPY

PART B—ISSUE FEE TRANSMITT

2426645
8/1-20

MAILING INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE. Blocks 2 through 6 should be completed where appropriate. All further correspondence including the Issue Fee Receipt, the Patent, advance orders and notification of maintenance fees will be mailed to addressee shown in Block 1 unless you direct otherwise by: (a) specifying a new correspondence address in Block 3 below; or (b) providing the PTO with a separate "FEE ADDRESS" for maintenance fee notifications with the payment of issue Fee or thereafter. See reverse for Certificate of Mailing, below.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending on the needs of the individual case. Any comments on the amount of time required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, D.C. 20231.

DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Box Issue Fee, Assistant Commissioner for Patents, Washington D.C. 20231

1. CORRESPONDENCE ADDRESS — 24M1/0619

WEINGARTEN SCHURGIN GAGNEBIN & HAYES
TEN POST OFFICE SQUARE
BOSTON MA 02109

AUG 15 1997

2. INVENTOR(S) ADDRESS CHANGE (Complete only if there is a change)

INVENTOR'S NAME

Street Address

City, State and Zip Code

CO-INVENTOR'S NAME

Street Address

City, State and Zip Code

☐ Check if additional changes are enclosed

APPLICATION NO.	FILING DATE	TOTAL CLAIMS	EXAMINER AND GROUP ART UNIT		DATE MAILED
08/318,252	10/05/94	017	LEWIS, C	2307	06/19/97

First Named Applicant: BACLAWSKI, KENNETH P.

TITLE OF INVENTION: DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD

ATTY'S DOCKET NO.	CLASS-SUBCLASS	BATCH NO.	APPLN. TYPE	SMALL ENTITY	FEE DUE	DATE DUE
2 NUB60XX	395-605.000	031	UTILITY	YES	$645.00	09/19/97

3. Correspondence address change (Complete only if there is a change)

09/23/1997 LNERGER 00000030 08318252
01 FC:142 645.00 OP
02 FC:561 30.00 OP

4. For printing on the patent front page, list the names of not more than 3 registered patent attorneys or agents OR, alternatively, the name of a firm having as a member a registered attorney or agent. If no name is listed, no name will be printed

1 Weingarten, Schurgin, Gagnebin & Hayes LLP
2
3

5. ASSIGNMENT DATA TO BE PRINTED ON THE PATENT (print or type)

(1) NAME OF ASSIGNEE: Northeastern University
(2) ADDRESS: (CITY & STATE OR COUNTRY) Boston, Massachusetts

A. ☐ This application is NOT assigned.

☒ Assignment previously submitted to the Patent and Trademark Office.

☐ Assignment is being submitted under separate cover. Assignment should be directed to Box ASSIGNMENTS.

PLEASE NOTE: Unless an assignee is identified in Block 5, no assignee data will appear on the patent. Inclusion of assignee data is only appropriate when an assignment has been previously submitted to the PTO or is being submitted under separate cover. Completion of this form is NOT a substitute for filing an assignment.

6a. The following fees are enclosed:
☒ Issue Fee ☒ Advance Order - # of Copies 10

6b. The following fees should be charged to:
DEPOSIT ACCOUNT NUMBER 23-0804
ENCLOSE A COPY OF THIS FORM
☐ Issue Fee ☐ Advance Order - # of Copies
☒ Any Deficiencies in Enclosed Fees

The COMMISSIONER OF PATENTS AND TRADEMARKS is requested to apply the Issue Fee to the Application Identified Above.

(Authorized Signature) Date: 8/14/97

NOTE: The Issue Fee will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the Patent and Trademark Office.

Certificate of Mailing

Note: If this certificate of mailing is used, it can be used to transmit the Issue Fee. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing.

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Box ISSUE FEE
Assistant Commissioner for Patents
Washington, D.C. 20231

on: 8/14/97 (Date)
Thelma E. Hawkins (Name of person making deposit)
Thelma E. Hawkins (Signature)
8/14/97 (Date)

PTOL-85B (REV. 05-96) Approved for use through 09/30/98. OMB 0651-0033 — 1. TRANSMIT THIS FORM WITH FEE — Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

JAR0002876

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Notice of Allowance dated: June 19, 1997
Issue Batch Number: 031

In re application	:	Kenneth P. Baclawski
Application No.	:	08/318,252
Filed	:	October 5, 1994
For	:	DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD
Examiner	:	C. Lewis
Attorney's Docket	:	NU-360XX

Group Art Unit: 2307

* *

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Box Issue Fee, Assistant Commissioner for Patents, Washington, D.C. 20231 on 7/2/97.

By _____
Stanley M. Schurgin
Registration No. 20,979
Attorney for Applicant

* *

RECEIVED
Publishing Division
JUL 07 1997
05

LETTER TO CHIEF DRAFTSMAN

RE: SUBMISSION OF FORMAL DRAWINGS

Box Issue Fee
Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Pursuant to the Notice dated June 19, 1997 submitted herewith in the above-identified application are formal drawings.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

Application No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2307

Kindly substitute these drawings for the ones currently on file as appropriate and charge any costs associated therewith to Patent and Trademark Office Account No. 23-0804. Triplicate copies of this transmittal letter are enclosed.

Respectfully submitted,

KENNETH P. BACIANSKI

By _____
Stanley M. Schurgin
Registration No. 20,979
Attorney for Applicants

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
Ten Post Office Square
Boston, Massachusetts 02109

Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: 7/2/97

Enclosure
SMS/rec
103988

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Notice of Allowance dated: June 19, 1997
Issue Batch Number: 031

In re application	:	Kenneth P. Baclawski
Application No.	:	08/318,252
Filed	:	October 5, 1994
For	:	DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD
Examiner	:	C. Lewis
Attorney's Docket	:	NU-360XX

Group Art Unit: 2307

* *

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Box Issue Fee, Assistant Commissioner for Patents, Washington, D.C. 20231 on 7/2/97.

By _____
Stanley N. Schurgin
Registration No. 20,979
Attorney for Applicant

* *

LETTER TO CHIEF DRAFTSMAN

RE: SUBMISSION OF FORMAL DRAWINGS

Box Issue Fee
Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Pursuant to the Notice dated June 19, 1997 submitted herewith in the above-identified application are formal drawings.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
TEL. (617) 542-2290
FAX (617) 451-0313

Application No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2307

Kindly substitute these drawings for the ones currently on file as appropriate and charge any costs associated therewith to Patent and Trademark Office Account No. 23-0804. Triplicate copies of this transmittal letter are enclosed.

Respectfully submitted,

KENNETH P. BACIAWSKI

By _____
Stanley M. Schurgin
Registration No. 20,979
Attorney for Applicants

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
Ten Post Office Square
Boston, Massachusetts 02109

Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: 7/2/97

Enclosure
SMS/rec
103988

FIG. 1

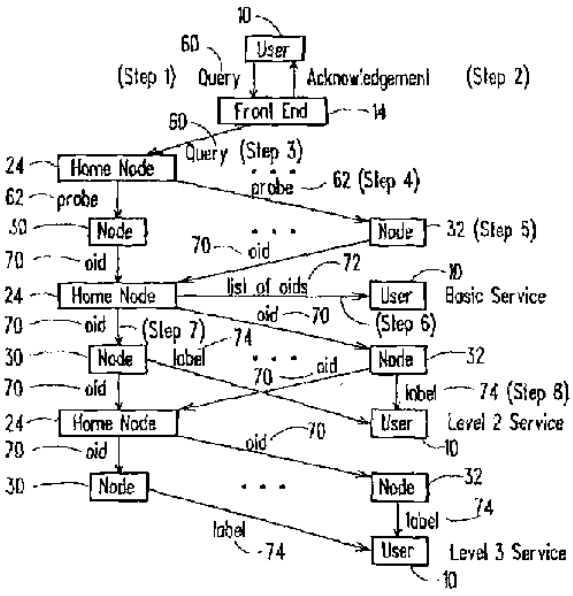


FIG. 2

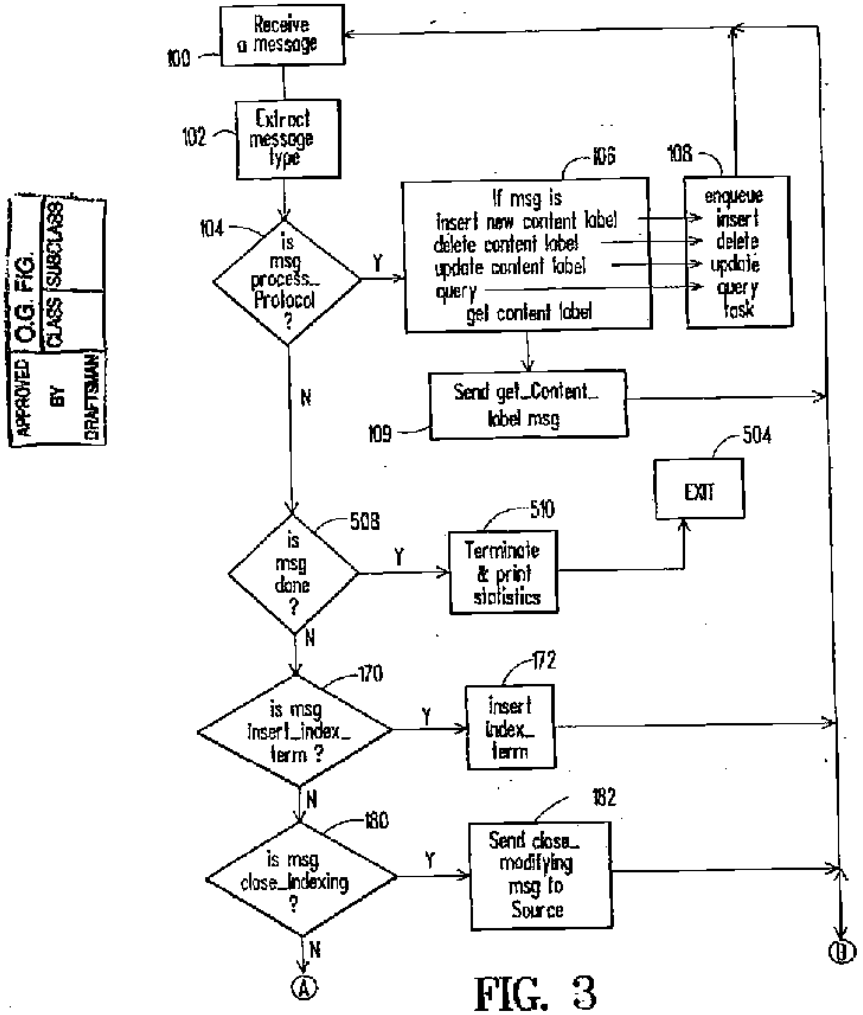


FIG. 3

O.G. FIG.
CLASS SUBCLASS
APPROVED BY DRAFTSMAN

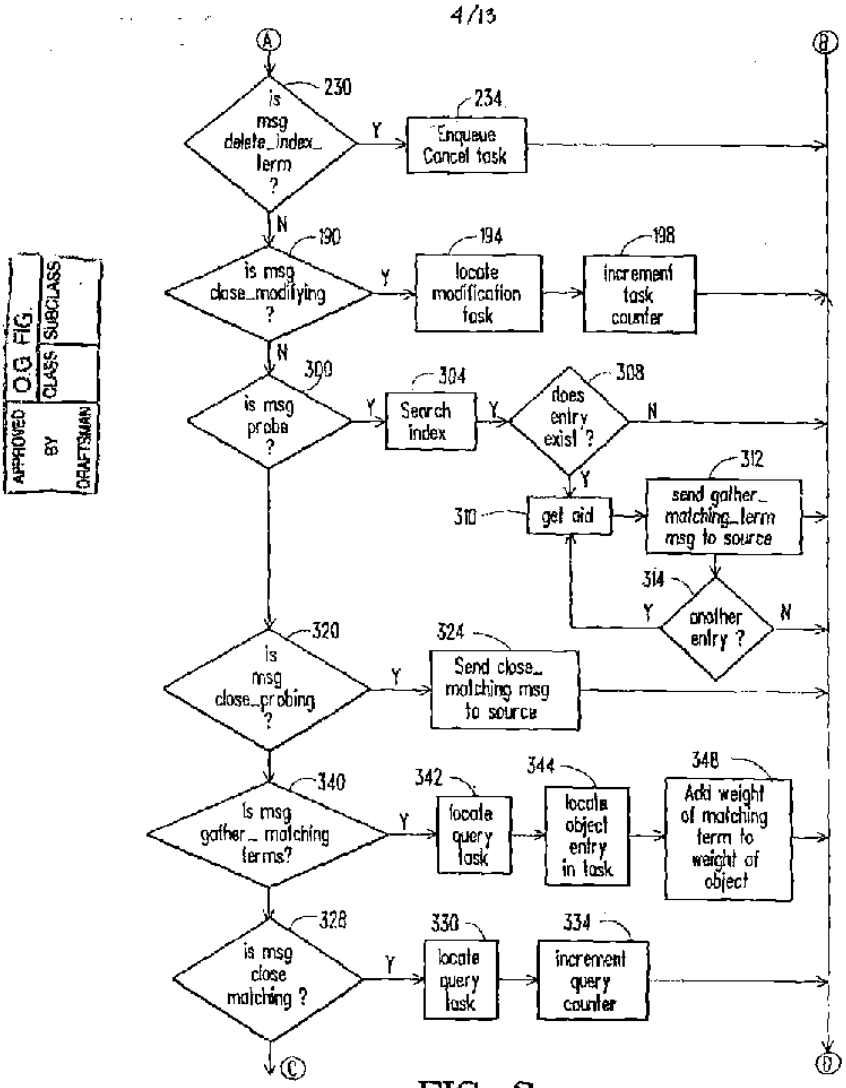

A
B
C
D
230
is msg delete_index_ term ?
Y
234
Enqueue Cancel task
N
190
is msg close_modifying ?
Y
194
locate modification task
198
increment task counter
N
300
is msg probe ?
Y
304
Search index
Y
308
does entry exist ?
N
Y
310
get aid
312
send gather_ matching_term msg to source
314
another entry ?
Y
N
320
is msg close_probing ?
Y
324
Send close_ matching msg to source
340
Is msg gather_ matching terms?
Y
342
locate query task
344
locate object entry in task
348
Add weight of matching term to weight of object
328
is msg close matching ?
Y
330
locate query task
334
increment query counter
APPROVED
BY
DRAFTSMAN
O.G. FIG.
CLASS
SUBCLASS

FIG. 3a

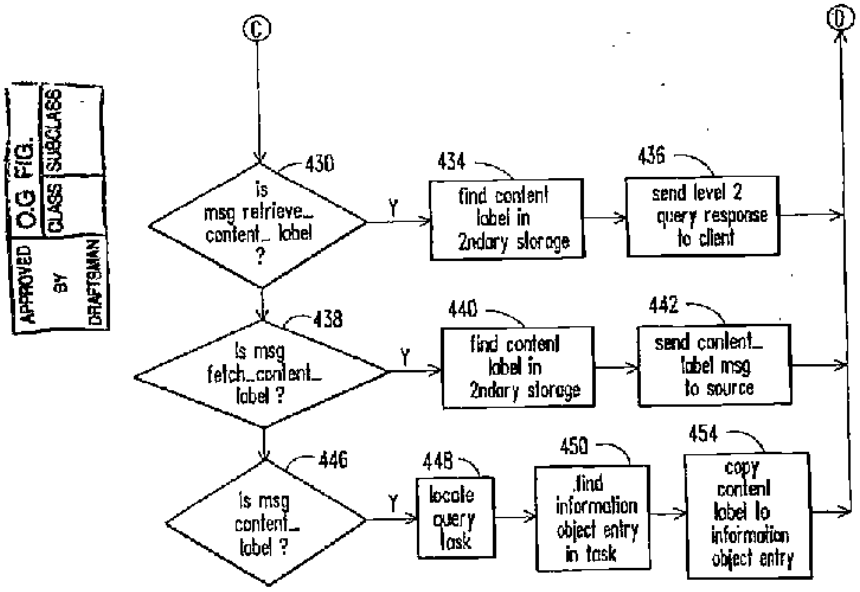

APPROVED BY DRAFTSMAN
O.G FIG.
CLASS
SUBCLASS
C
D
430
is msg retrieve_ content_ label ?
Y
434
find content label in 2ndary storage
436
send level 2 query response to client
438
is msg fetch_content_ label ?
Y
440
find content label in 2ndary storage
442
send content_ label msg to source
446
is msg content_ label ?
Y
448
locate query task
450
find information object entry in task
454
copy content label to information object entry

FIG. 3b

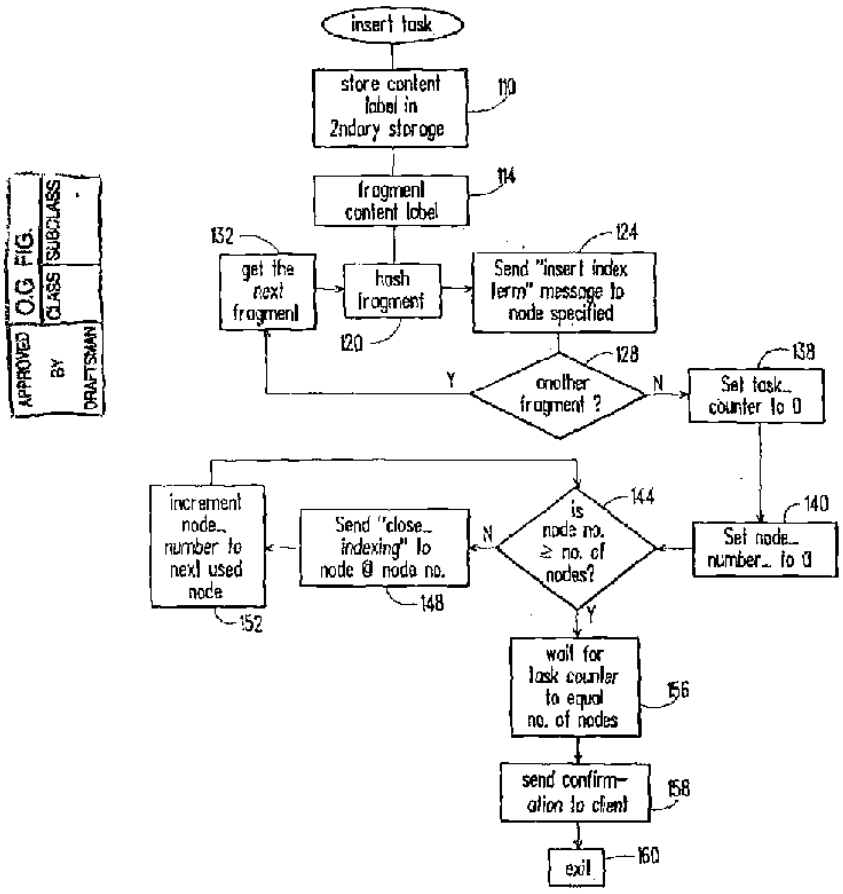

insert task
store content label in 2ndary storage
110
fragment content label
114
get the next fragment
132
hash fragment
120
Send "insert index term" message to node specified
124
another fragment ?
128
Y
N
Set task counter to 0
138
Set node number to 0
140
is node no. ≥ no. of nodes?
144
N
Y
Send "close indexing" to node @ node no.
148
increment node number to next used node
152
wait for task counter to equal no. of nodes
156
send confirmation to client
158
exit
160
APPROVED
O.G FIG.
BY
CLASS
SUBCLASS
DRAFTSMAN

FIG. 4

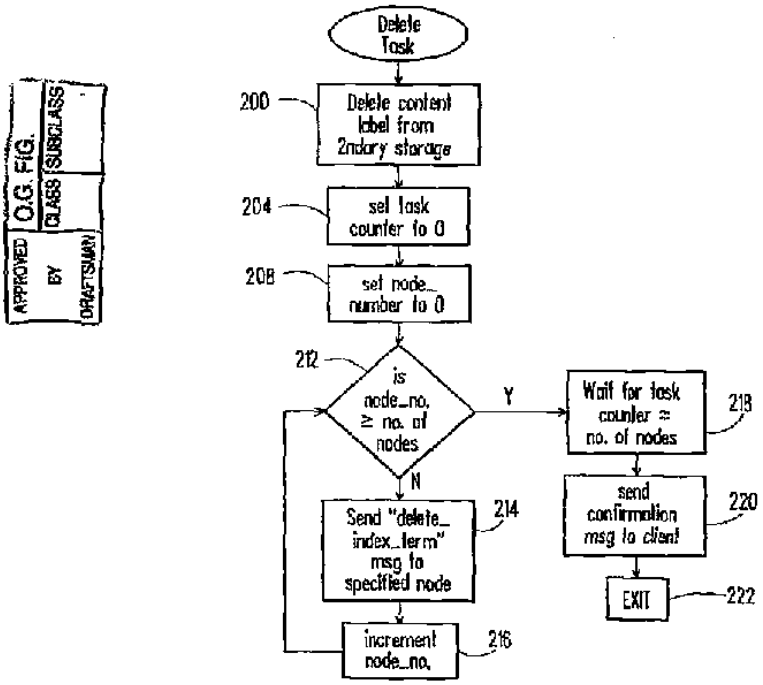

O.G. FIG.
CLASS
SUBCLASS
APPROVED
BY
DRAFTSMAN
Delete Task
200
Delete content label from 2ndary storage
204
set task counter to 0
208
set node_ number to 0
212
is node_no. ≥ no. of nodes
Y
N
Wait for task counter = no. of nodes
218
send confirmation msg to client
220
EXIT
222
Send "delete_ index_term" msg to specified node
214
increment node_no.
216

FIG. 5

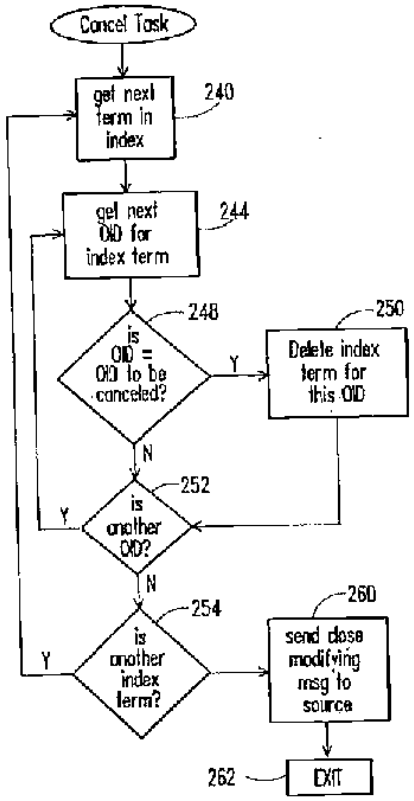


FIG. 6

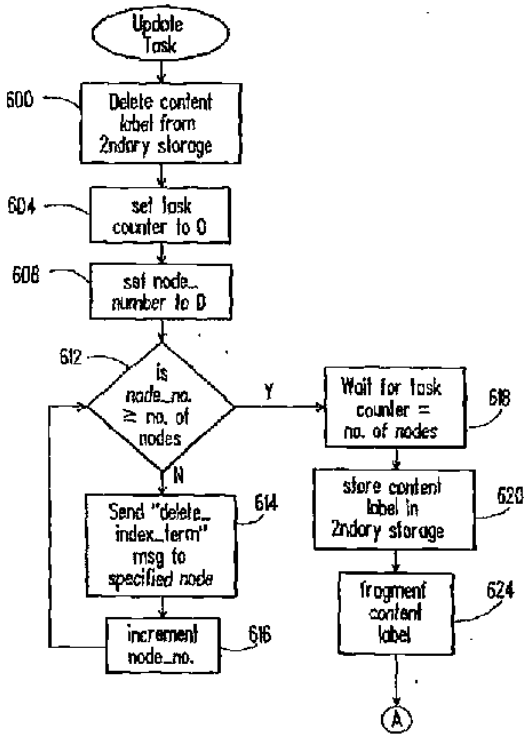


FIG. 7

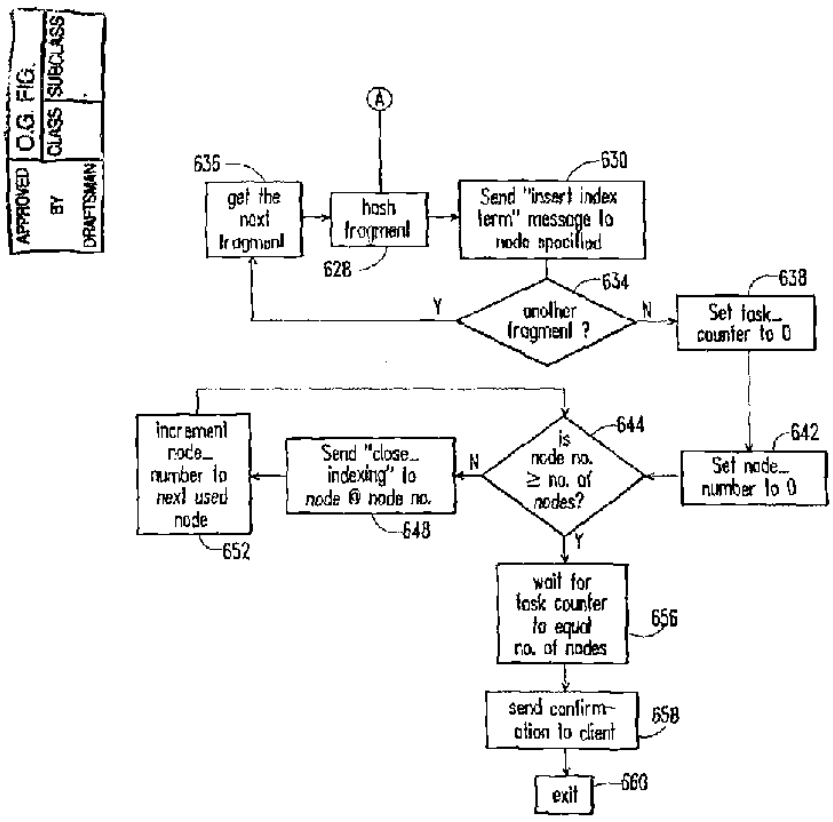

APPROVED BY DRAFTSMAN
O.G. FIG.
CLASS
SUBCLASS
A
635
get the next fragment
hash fragment
628
630
Send "insert index term" message to node specified
634
another fragment ?
Y
N
638
Set task_ counter to 0
642
Set node_ number to 0
644
is node no. ≥ no. of nodes?
N
Y
Send "close_ indexing" to node @ node no.
648
Increament node_ number to next used node
652
wait for task counter to equal no. of nodes
656
send confirm- ation to client
658
exit
660

FIG. 7a

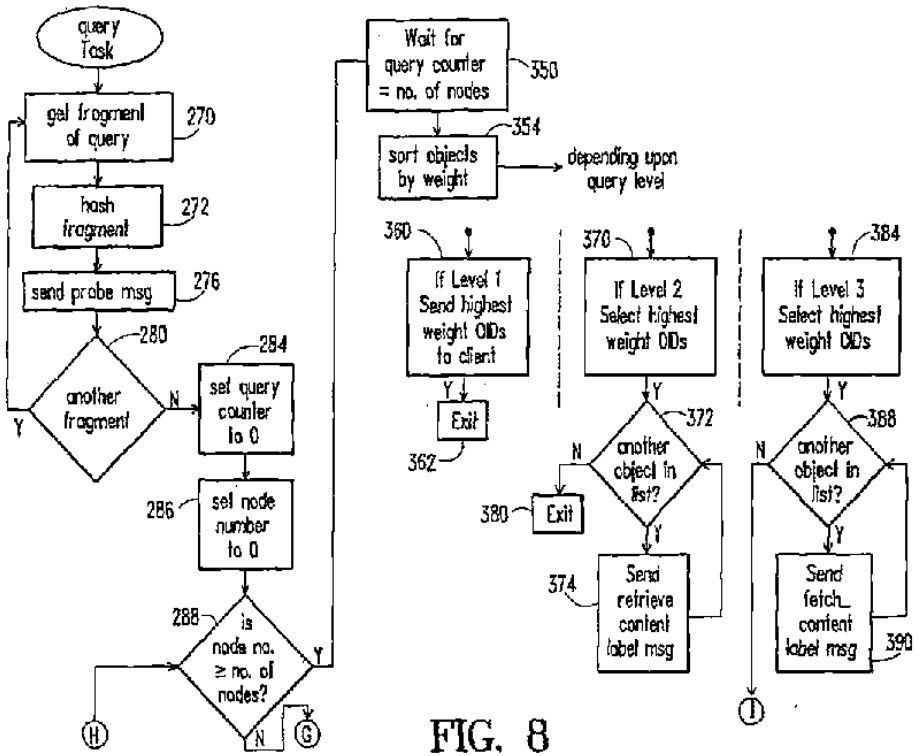


FIG. 8

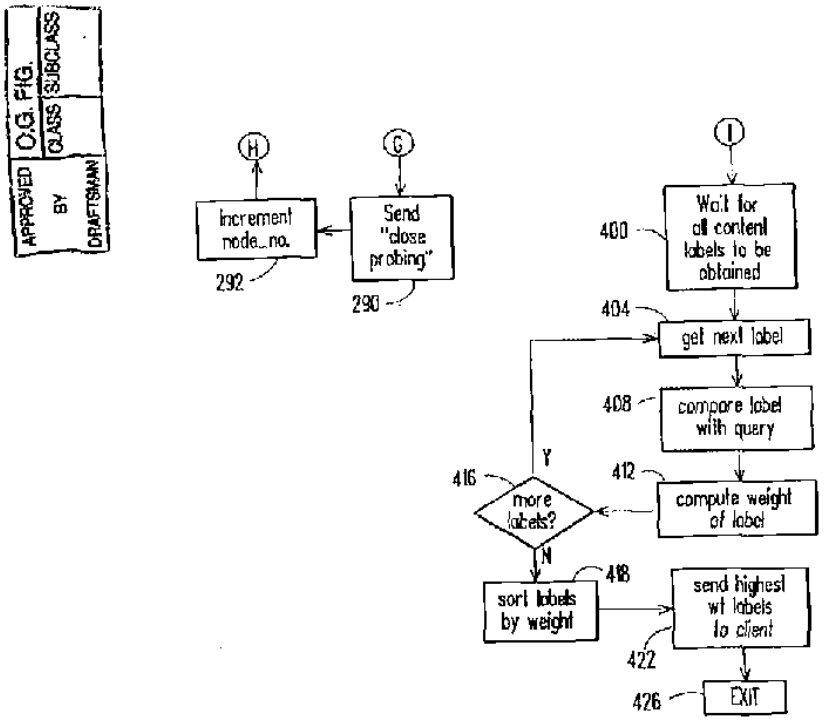

APPROVED
BY
DRAFTSMAN
O.G. FIG.
CLASS
SUBCLASS
H
G
I
Increment node_no.
Send "close probing"
Wait for all content labels to be obtained
get next label
compare label with query
compute weight of label
more labels?
Y
N
sort labels by weight
send highest wt labels to client
EXIT
292
290
400
404
408
412
416
418
422
426

FIG. 8a

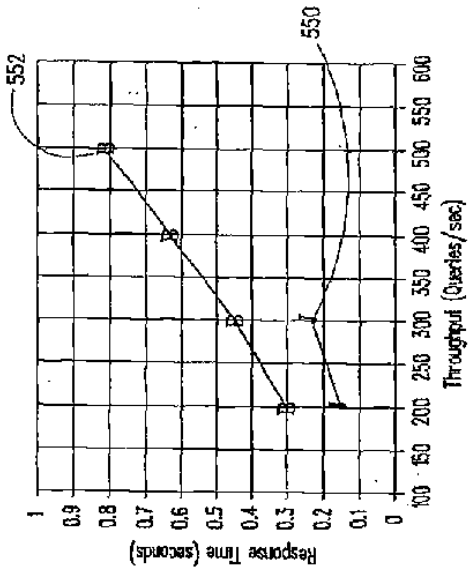


FIG. 9

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Notice of Allowance dated: June 19, 1997
Issue Batch Number: 031

In re application	:	Kenneth P. Baclawski
Application No.	:	08/318,252
Filed	:	October 5, 1994
For	:	DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD
Examiner	:	C. Lewis
Attorney's Docket	:	NU-360XX

Group Art Unit: 2307

* *

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Box Issue Fee, Assistant Commissioner for Patents, Washington, D.C. 20231 on 7/2/97.

By _____
Stanley M. Schurgin
Registration No. 20,979
Attorney for Applicant

* *

LETTER TO CHIEF DRAFTSMAN

RE: SUBMISSION OF FORMAL DRAWINGS

Box Issue Fee
Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Pursuant to the Notice dated June 19, 1997 submitted herewith in the above-identified application are formal drawings.

WEINGARTEN, SCHURGIN, GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

Application No.: 08/318,252
Filed: October 5, 1994
Group Art Unit: 2307

Kindly substitute these drawings for the ones currently on file as appropriate and charge any costs associated therewith to Patent and Trademark Office Account No. 23-0804. Triplicate copies of this transmittal letter are enclosed.

Respectfully submitted,

KENNETH P. BACLAWSKI

By _____
Stanley M. Schurgin
Registration No. 20,979
Attorney for Applicants

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
Ten Post Office Square
Boston, Massachusetts 02109

Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: 9/2/97

Enclosure
SMS/rec
103988

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

C9C

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application	:	Kenneth P. Baclawski
Patent No.	:	5,694,593
Issued	:	December 2, 1997
For	:	DISTRIBUTED COMPUTER DATABASE SYSTEM AND METHOD
Attorney's Docket	:	NU-360XX

314252

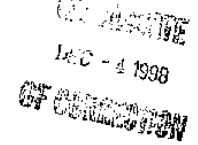
* *

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on 12/3/98.

BY _____
Stanley M. Schurgin
Registration No. 20,979
Attorney of Record

* *

LETTER



Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

In proofreading the above-identified patent as printed, the following errors were noted:

Column 2, line 51, "Figs. 7 and 7a is" should read --Figs. 7 and 7a are--.

Column 6, line 11, "delete" should read --deleted--.

Column 7, line 35, "If the is another fragment" should read --If there is another fragment--.

Column 8, line 42 and 43, "precomputered" should read --precomputed--.

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

Patent No: 5,694,593
Issued: December 2, 1997

Column 10, line 21 and 22, "Therefore, it is the intention no limit" should read --Therefore, it is the intention to limit--.

Since these errors do not appear to affect the sense of the text, a Certificate of Correction is not believed necessary. It is requested, however, that this letter be included with the official application file for record purposes.

Respectfully submitted,

KENNETH P. BACLAWSKI

By _____
Stanley M. Schurgin
Registration No. 20,979
Attorney of Record

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
Ten Post Office Square
Boston, Massachusetts 02109

Telephone: (617) 542-2290
Telecopier: (617) 451-0313

Date: 12/3/98

SMS/laf
132093

WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

PTO UTILITY GRANT

Paper Number 11

The United States of America

The Commissioner of Patents and Trademarks

Has received an application for a patent for a new and useful invention. The title and description of the invention are enclosed. The requirements of law have been complied with, and it has been determined that a patent on the invention shall be granted under the law.

Therefore, this:

United States Patent

Grants to the person(s) having title to this patent the right to exclude others from making, using, offering for sale, or selling the invention throughout the United States of America or importing the invention into the United States of America for the term set forth below, subject to the payment of maintenance fees as provided by law.

If this application was filed prior to June 8, 1995, the term of this patent is the longer of seventeen years from the date of grant of this patent or twenty years from the earliest effective U.S. filing date of the application, subject to any statutory extension.

If this application was filed on or after June 8, 1995, the term of this patent is twenty years from the earliest effective U.S. filing date of the application, subject to any statutory extension.

Bruce Lehman

Commissioner of Patents and Trademarks

Attest

Form PTO-1584 (Rev. 5/96)

(RIGHT INSIDE)