The $\mathcal{IQ}$ System[⋆] : A Deductive Database Information Lens for Reasoning about Textual Information

Bob Kero and Shalom Tsur

Argonne National Laboratory
Decision and Information Sciences Division
Argonne, Illinois, 60439
{bkero, tsur}@dis.anl.gov

Abstract. The $\mathcal{IQ}$ system, described in this paper, addresses two current problems in information retrieval (IR) from document repositories: (i) the "glut-of-information" that threatens to overwhelm the users of these systems and (ii) the loss of informational value as the documents are moved from their organizational setting into a machine-readable format. The answer to these problems discussed in the paper, proposes to restore much of this information by creating a context-base to store the additional recovered knowledge. The technology used is the $\mathcal{LDL}$++ deductive database: we demonstrate how to use the rule-extensions to declare levels of abstraction in the information that can be used to "zoom-in" or "pan-out" as required by the user (an "Information Lens"). The system takes extensive advantage of the World-Wide Web and enlists the capabilities of JAVA for information display. We discuss some of the performance issues that were experienced and conclude with a description of future R&D ideas along the lines started here.

1 Introduction

Many government agencies, private companies and other large organizations maintain or are developing an "Information Center" as part of their information infrastructure. The role of the Information Center within the organization is not unlike that of a library: to serve as a central repository for internal documentation and information received from external sources. The incoming information is archived and in addition, serves for day-to-day decision making purposes.

The rate of full-text information intake at the Department of Energy (DOE) Environment, Safety, and Health (ES&H) Information Center, as in most Information Centers, is constantly increasing. Most of this information is received in the form of scientific and technical documents on paper or in electronic format. In its present form, this ever-increasing flow poses a "glut-of-information" problem to the decision maker, who is forced to sift through it to select relevant pieces

---

[⋆] Work supported by the U.S. Department of Energy, Office of Information Management (EH-72), contract W-31-109-Eng-38.

of information. The fact that most decisions need to be taken in some limited time compounds the problem. Typically, it becomes impossible to review all of the potentially relevant documentation within the time frame allotted to decide. The present form of full-text searching support raises thus the serious concern that the quality of decision making may be impaired because decisions may be based on incomplete or random knowledge. This problem will be exacerbated as the present economic climate and tightening budget constraints force many organizations to do more with less.

Besides the information-glut problem, another factor that also contributes to the present unsatisfactory state of affairs is the *decontextualization* of the stored electronic full-text documents. This term deserves an explanation. Briefly, documents are created for a specific reason and within a specific context. The reason is often to meet a legislative or policy requirement, while the context is based on the organization(s) involved, people who created the document, time and place of authorship, and relationships to other documents and issues. For example, a DOE "Finding of No Significant Impact" document possesses at least one closely coupled specific relationship, lineage: child to parent, to its respective environmental assessment document. This type of additional knowledge, often vital to the decision-making process, is lost or made difficult to discern when each document's textual content is all that is preserved. Presently, the burden of remembering (or the cost of dynamically determining) contextual information is the sole responsibility of each user. This arrangement can quickly result in a heavy economical cost to the organization. Domain experts leverage their individual a priori knowledge of the corpus and its contents to better control their information searching and gathering. Novice users drown in what falsely looks like a predominately unrelated sea of documents. In either case, there is no capitalization upon an organizationally defined contextual understanding of the corpus, which would (1) increase searching efficiency, and (2) reduce uncertainty in the decision-making process for all classes of users. Instead, present technology normally allows the retrieval of the documents only if certain terms that appear in their text are used. The context is not an explicit part of the documentation; hence, it cannot be used in the search for relevant documentation. Consequently, the decision maker who must make decisions solely on the basis of document contents, without understanding their context, can be at a huge disadvantage.

The objectives of the $\mathcal{IQ}$ system are to effectively address these issues. The approach that we adopted was to *augment* the already existing information with contextual knowledge kept in a separate *context base*—a deductive database implemented in $\mathcal{LDL}$++ [AOTZ]. The responses to user queries, returned from IR engines containing documentation sources, are elaborated using the context base and distributed using the World-Wide Web (WWW) in the form of JAVA applets[2] [GM95]. Section 2.0 describes the evolution and architecture of the

---

[2] JAVA, is an architecturally neutral object-oriented programming language developed by Sun Microsystems, Inc. Applets, JAVA programs, can be transmitted over the Internet, run on multiple clients, and provide users with the multimedia richness of a CD-ROM.

$\mathcal{IQ}$ system. Section 3.0 describes the documentation reengineering process necessary to take advantage of this approach. This includes a reengineering of the existing stored documentation, and a means to encode future documentation to capture the contextual knowledge. Section 4.0 describes the technical considerations and experience gained thus far with the use of deductive database technology as embedded in this system. We conclude with some ideas for the next generation of the $\mathcal{IQ}$ system, briefly describe the cost-benefit considerations of the reengineering process and compare it with the conventional method of handling documentation.

2 The $\mathcal{IQ}$ Architecture

The overall goal of developing the $\mathcal{IQ}$ system is to create a tool to provide not only access, management, and storage of the information content found in the DOE ES&H Information Center, but also to the rich contextual tradition surrounding its uses in particular DOE communities-of-practice. Objectives of the system include providing a means to:

- retain the value of information content and its uses
- capture the rich contexts associated with information content as new information.
- coherently visualize large collections of information content through graphic abstractions and classifications
- contextually understand the relationships between hit list[3] elements and their associated documents
- interact with both document content and context
- allow for the definition and integration of future information contexts
- control the rate and amount of information content transmitted to a user

When the system is fully developed it will provide an effective means to manage the adaptation of the documents' content, its context, and its changing needs over time. Additionally, it will reduce the risk of decision making by providing accurate, timely and complete information to decision makers through a recontextualization process (i.e., the partial or full integration of information content and context). Finally, $\mathcal{IQ}$ will continually reduce the difficulty of managing and comprehending the accumulation of information content as it grows rapidly in volume and diversity.

The architecture of the $\mathcal{IQ}$ System, see Figure 1, is composed of three major pieces:

1. *The Document Repository.* This piece contains all of the document content found in the system.

---

[3] *Hit list* – a relevancy ranked list of document or document component identifiers (e.g., titles) corresponding to the textual content that the IR system deems pertinent to the users' query.

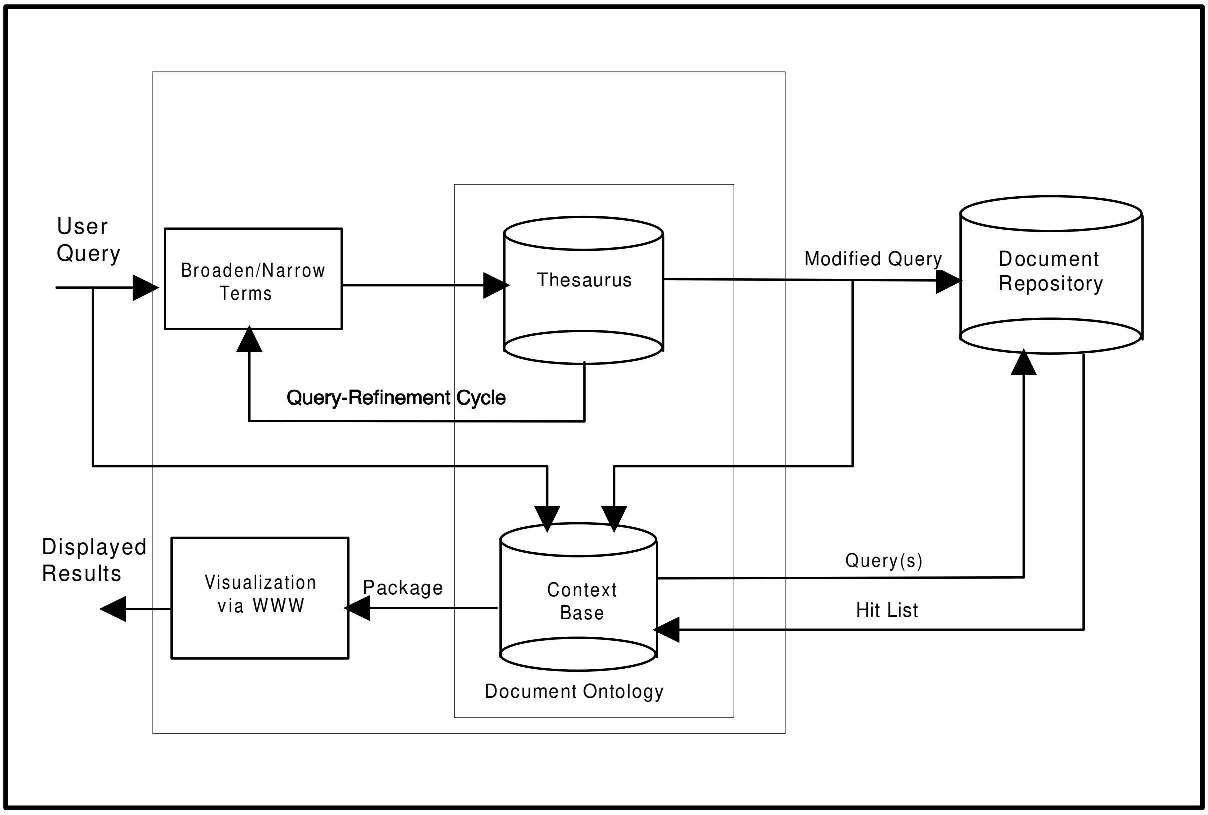

Fig. 1. $\mathcal{IQ}$ System Architecture

2. *The Document Ontology.* This piece contains knowledge pertaining to the relationships between pieces, whole, or classes of documents kept in the document repository, the linguistic knowledge describing and relating the uses of terminology found in those documents, and the rules governing the uses of all types of knowledge.
3. *The User Interface.* This piece allows users to access (e.g., query or view from different perspectives) the information in the document ontology and the document repository. Users can observe the results of their queries in various abstract graphical representations, instead of in hit lists, that will provide them with different contextual understandings. The interface visually abstracts the information using filters so users can reason with its specific contexts instead of just scanning or reading its actual text-level contents. This extensible set of information filters is used to constrain the information while the system's user controlled *information lens* is allowed to zoom in and out as to provide more or less detail.

The architecture of the $\mathcal{IQ}$ system is intrinsically connected to the use of information retrieval technologies, yet it is not tightly coupled to any particular product or method. Instead, the $\mathcal{IQ}$ system has been designed to augment current and future IR systems by building a uniform cap over them. Before providing an overview of the technology we will put the use of IR systems in a perspective. The advent of the World-Wide Web has brought a strong resurgence in the need to provide better techniques to access information. As enormous amounts of textual information continue to be loaded onto the WWW at unprecedented rates, a refocusing on the strengths and weaknesses of the current state of IR has occurred. At this time the field of IR is rapidly advancing. Innovations continue to be added to public domain and commercial IR systems at an accelerated pace, which makes choosing a specific product for a particular organization very difficult. Our approach should be seen as a way to lessen the risk of selecting IR systems.

2.1 Overview of the Technology for Information Retrieval

Information Retrieval technology is used to facilitate the retrieval of a subset of the relevant documents to a user, from a corpus of stored documents. Typically the assumption is that the corpus itself is large and can reach the order of hundreds of thousands of documents and the user specification pertains to a small subset of these, maybe on the order of tens of documents at most. We use the term “document” in a generic sense to refer to such magnetically or optically stored items as reports, manuals, newspapers, magazines, etc. Sometimes, when the documents are voluminous, the actual document is kept in its original paper format and only a reference to it is kept as part of the corpus. For a short discussion of the users requirements in this area see [Cro95].

The user specification is submitted to the system in form of a query, and the system's response is returned in the form of a hit list. In this paper we will distinguish between “traditional” methods for information retrieval and “knowledge intensive" methods. The former ones use the document texts, as they appear in the corpus, as the sole source of information for the resolution of the user's query; the later use additional information in the form of thesauri and other linguistic information to enhance and improve the query resolution process. Finally, we will discuss how the $\mathcal{IQ}$ system extends the abilities of current IR systems through a complementary design.

2.2 Traditional Methods for Full Text Searching

Traditional IR systems such as WAIS [WAI95] and Harvest [HAR95], store the full text of the document and group sets of documents in a corpus. These systems provide support for extensive indexing methods that allow for the retrieval of documents relevant to a user-posed query. User-queries are formulated as conjunctions, disjunctions and other Boolean expressions using terms being searched for in the corpus. The result of a query is in the form of a relevancy ranked hit-list.

The basic limitation of the traditional information retrieval methods is that, in order to access a document, its text must contain the *exact* terms that were used in the query. Thus, if a query uses the term "automobile," whereas the document contains the term "car," it will not be included in the hit list. This is also true for misspelled words, British vs. the American use of the language, the use of slang, etc. Another limitation is that all that is returned by a hit list, is the fact that a certain document contains a query term. Little is revealed about the distribution of the term throughout the document, or if the term was used in the title, body, or reference section of the document. This knowledge is often very important in assessing its relevance. The statistical measures typically employed contribute only very little in this respect since all they do are count occurrences but do not reveal anything about the whereabouts of the words in the text.

The measures of success typically used with IR technology include *precision* and *recall* which respectively, register the extent to which those documents appearing in the hit list should have been included and the extent to which all of the relevant documents have been hit. Over the years, the methods used for the resolution of user queries have continuously been refined to achieve high degrees of these measures.

2.3 Knowledge-Intensive Methods for Information Retrieval

Current commercial IR products such as Personal Librarian by Personal Library Software Corporation [PLS95] and RetrievalWare by Excalibur Technologies [EXC95], offer a range of knowledge-intensive capabilities. For example, RetrievalWare offers a high-end semantic network containing over 400,000 word meanings and over 1.6 million relationships to assist its users in query development and searching. The current $\mathcal{IQ}$ system has foregone the inclusion of general purpose dictionaries and thesauri and instead, has concentrated on the inclusion of one DOE domain specific dictionary/thesaurus on the order of 30,000 terms. This specialized dictionary/thesaurus consists of the approved vocabulary used throughout specific areas in DOE and contains many inter-term relationships including: broader term, more narrow term, related term, see also, see for, use in place of, used for. The vocabulary found in this resource permeates the ES&H corpuses of interest and therefore provides a valuable resource to the DOE ES&H Information Center's user base.

The following example provides a general understaing of how various inter-term relationships can be used in the query development and refinement process. Interaction with a thesaurus enables the *narrowing* or *broadening* of potential query terms. Thus the terms "Cardiologist," "Endocrinologist," and "Urologist," could be broadened to "Medical Doctor" and substituted in the query. Alternatively, a general term can be replaced with more specific instances. This process is driven by the structure and content of the thesaurus used.

2.4 Advancing the State of the Art

Traditionally, the user-system interaction is characterized by the following cyclic behavior:

```
until satisfied do
    send query to document source;
    receive hit list;
    read contents of each item on hit list;
    modify (or formulate new) query;
od
```

At each repetition of the cycle, the user scans the textual information in the hit list and based on this new input, decides whether to modify his/her query until the final hit list is deemed to be appropriate to his/her needs. This process, which is the textual equivalent of the "data mining" process in numerical databases, is tedious and as we already stated, is often an impediment in effective decision making. In contrast, the user interaction with the $\mathcal{IQ}$ system is of a very different nature. The user controls *two* cycles of interaction: one cycle to narrow or broaden the search terms used in each version of the query and the other cycle, using the context base, to observe the results at different levels of abstraction. This process can be likened to "zooming" and "panning" in and out of the information through an information lens. Only at the last stage, at the highest level of resolution, does the user actually access the textual information of the corpus. In some cases this last step may be entirely omitted! We expect the interaction with the meta information, rather than with the actual text to be vastly more efficient and to contribute in a significant way to the timeliness and quality of decision making. From the performance perspective, all of the visualization capabilities of the system are on the client side. Thus, the volume of information during each of the phases of the interaction is much smaller than the interaction with the document repository which necessitates the actual transfer of a set of documents from the server to the client.

3 Document Preparation

In this section we provide an overview of the mark up process and outline the steps that must be taken to transform existing and future corpuses of documentation into a format that lends itself to the $\mathcal{IQ}$ architecture.

3.1 Document Mark up

Organizations normally deal with diverse sets of documents which vary in size. When the size of documents grow beyond a few pages, the documents need to be decomposed in a way (e.g., logically by sections or paragraphs) suited to more efficient searching. For example, a user searching for a particular term would normally want to receive the immediate section of information surrounding the term, instead of the complete document the term was found in. To decompose a document into its component parts it is first necessary to identify the boundaries of each component. Since most documents are used in different ways and in different information systems throughout their life cycle, it is highly efficient to insert neutrally-formatted mark up (tags), based on each component's type, within the document to identify its components for future uses. After a document has been marked up, tools can be used to physically split and transform its content, based on the included tags, into the formats necessary to be used in commercial IR systems. Establishing a neutral approach allows for organizations to pay for the document preparation only once, even if information systems and their subsequent formats are changed in the future. The minimal cost of developing transformation filters from the neutral format to any new specific format would be the only future re investment ever needed. Documents that were produced using word processing systems (e.g., Microsoft Word), or other mark up languages (e.g., TeX), already contain an internal mark up. Native mark up formats can be used directly or can be transformed into an equivalent mark up language standard set of tags.

The mark up language of choice for the $\mathcal{IQ}$ system is the Standard Generalized Mark up Language[4] (SGML). One of the strengths of SGML relates to its ability to mark up a document in a neutral way, independent of the many ways the document can be processed in the future. [Smi92] The tagged documents must conform to a DTD[5]. An SGML parser (e.g., SGMLS) can be used to infer the structure of the document through the use of its included tags and check to see whether it is compatible with its declared DTD. The result of the document preparation process is a collection of document components, and a unique form of an enhanced parse tree used to preserve the logical structure and physical layout relationships between document components within a document.

---

[4] *SGML* – an international electronic document exchange standard developed by the International Organization for Standardization (ISO)

[5] *Document Type Definition*— a specific set of document component types, their relationships, the tag set used for mark up, and a set of rules for which the presence (or absence) of document components at particular times could be checked.

It should be noted that the HyperText Mark up Language (HTML) is defined in an SGML DTD. In essence, HTML is a collection of styles (indicated by mark up tags) that define the components of a WWW document. The $\mathcal{IQ}$ system dynamically transforms the potentially more robust SGML marked up documents and document components into HTML for preview.

3.2 Document Preparation Process

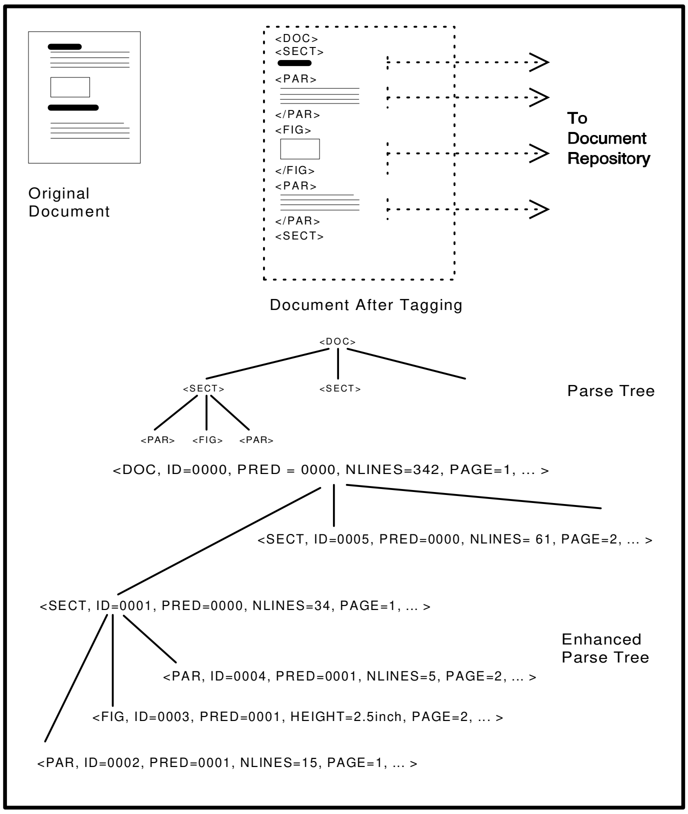

Fig. 2. Document Processing Steps

Figure 2 graphically depicts the steps a document must go through to construct its enhanced parse tree.

Step 1: *Marking-up of the documentation.* During this step a document in ASCII format is marked-up using the SGML standard. Tags are assigned to each of the documents' components. This step, which clearly is the most labor-intensive in the process, can be performed either manually, meaning that the person(s) performing this task use a tool such as a general-purpose editor to scan the document, identify the components and insert the appropriate tags, or automatically, using a custom mark up tool or a commercial one such as Avalanche's FastTag, which identifies the components and attaches the tags to them.

Step 2: *Parsing of the marked up documents.* The tagged documents must conform to a DTD. A parser such as SGMLS infers the structure of the document and checks to see whether it is compatible with the declared DTD. The result of this operation is the parse tree and the document components that correspond to each of the nodes of this tree. Incompatible documents are rejected.

Step 3: *Enhancement of the parsed document.* At this time each document component is assigned a unique id, and the same id is assigned to its corresponding tag instance in the parse tree. The tags are then enhanced with additional information attributes. Attributes pertaining to a document include, but are not limited to, the page number on where each document component starts, the size of the component (e.g., in lines), a link between it and its parent component, the type of the component, the offset from the top of the first page its starts on, and other ancillary information (e.g., titles or captions).

Step 4: *Splitting of the information.* Once enhanced, the textual parts and their attached id's are sent to a document repository (i.e., the IR engine) and the enhanced tags are sent to the context base. Thus, from the perspective of the repository, the input is still in its native format except that in actuality, each record in the repository is now a component of an original document. This fine granularity of information is of no consequence to the repository which should treat each of these records as if they were autonomous.

The information for the context base consists of tuples which can be stored in a standard, relational format. The context base can therefore be realized using an off-the-shelf relational database.

4 Context Base: the Role of Deductive Database Technology

In this section we discuss some of the details and the experience gained so far with the context base of the system. The purpose of the context base is, as we suggested, to store knowledge *about* the documentation as opposed to the knowledge contained in the documentation itself. We distinguish between *inter* and *intra-document* knowledge among documents. Intra-document knowledge pertains to relationships found within each of the documents as separate entities. For example, the relationships described between the various document components: paragraphs, sections, chapters etc. are part of the document's intra-document knowledge. This knowledge, which is extracted from the documents as parse-trees, is stored in an enhanced form in the context-base using the process described in the previous section. Thus, the requirement of efficient traversal of the forest of trees kept in the context base becomes of paramount importance in the implementation of the system and has, to a large extent, motivated us in the choice of deductive database (DD) technology for the project.

Inter-document knowledge pertains to relationships among separate documents in the document repositories. Typical relationships include common geographical sites or institutions at which documents originated, temporal relationships among documents and other relationships, of particular importance to individual users of the system. We elaborate on this subject later in this paper.

A third component of the context-base is the Thesaurus. As mentioned in Sec. 2, it is used to narrow or broaden the search terms presented to the document server. In our case the Data for the Thesaurus were given and these determined a set of relations such as `broader_term`, `narrower_term`, `definition`, `citation`, `synonym`, and `scope`. These relations can be queried to assist the user in the formulation of the right query to the document sources. Presently, the volume of the Thesaurus is on the order of 30,000 terms.

Another aspect that warrants in our view the use of DD technology is *Management of Change:* this problem needs to be examined from both the system administration perspective as well as the individual users'. The glut of information problem that we mentioned in the introduction forces the Information Center administration to regulate and control not only the intake of documents but also the outflow: documents that have become outdated or whose content is not relevant anymore to the present decision-making needs. The individual user typically responds to changing policies or other events and hence his/her needs rapidly change over time. As needs change, the capability to rapidly update and modify the views of the information presented to users must be available. We believe that DD technology can provide the flexibility combined with acceptable performance to accommodate this rapid change.

4.1 Logical Organization of the Context Base

Table 1 depicts the logical organization of the context base. It is divided into 5 sections:

1. Intra-document relations. This section contains the schema information for the `intra` relation containing the parse trees for the documents. Each component node has its own ID and the tree is represented by a parent-child relationship between nodes. `intra` contains also the additional information extracted during the parsing such as the physical layout of each node of the parse tree (page number, number of lines) used to generate the "display package" sent to the client viewer via a JAVA applet.
2. Inter-document relations. Presently, this section contains the `doc` relation which contains a record for each document with global information about it: origins, date of creation, title and other information.
3. Thesaurus information. This section contains the relation declarations required for the thesaurus. Some of these relations were mentioned earlier in this section.

4. Expanded information. The expanded information is required to augment the intra-document knowledge derived from the parse process. It contains derived, aggregate information about each of the documents. For example, for each node of the parse tree it contains the aggregate subtree size and range of node id's for the subtree of which the node is the root. This information, which is required for display purposes, can only be derived once the parse tree is materialized in the `intra` relation[6].
5. Display rules. Once the document information is collected and expanded, then a query form, compiled from the display rules, determines which portion of the information should be packaged and sent to the client for viewing. The portion is hit-dependent. We elaborate on this topic in the next section. Other rules are used to form query forms that can be invoked by the user to respectively request more or less information.

sec.	*function*	*relations*	*size*
1	intra-document	`intra`	≈2000 records/doc
2	inter-document	`doc`	1 record (≈200 bytes)/doc

3	thesaurus	`broader_term`
		`narrower_term`
		`definition`
		`citation`
		`synonym`
		`scope`
			30,000 terms

4	expanded	`stats`	one record for each `intra` record
5	display	`display rules`	22 rules (incl. book and house-keeping rules)

Table 1. Logical organization of the context base

4.2 Display and Interaction with Relevant Information

Figure 3 depicts the view of a document when seen by the user through a web browser[7]. In the lower part of this view a sequence of "thumb nails," each representing a page of the underlying document, can be seen. The thumb nails cannot be read but they contain sufficient information to form a birds'-eye (meta) view that reveals the informational outlines of the document and allows the user to

---

[6] In principle, this information could be collected "on the fly," during the parsing process. We do not however have any control over this process, nor do we know whether it is bottom-up. Therefore this option does not exist in our case.

[7] In the actual display, the bars shown in different grey shades in the picture, are different colors, representing the relevancy of the documents in the query result.

Fig. 3. Document Meta-view

form an opinion as to the actual relevance of the underlying information. The user can zoom in on the details by clicking on a thumbnail; thus causing the retrieval of the actual textual component containing the page. Each of the sections, graphs, tables and diagrams on a page can be discerned at the meta-level: sections of the document that were hit i.e., contain terms that were included in the user-query, have a different color which is determined by a relevance ranking formula. The user can, for example, easily observe the density of hits without having to retrieve the document itself.

Not all of the pages of the underlying document are represented at the same level. Portions of the document that are deemed irrelevant to the user-query are aggregated into "piles," represented by special, stack-like thumb nails. Clicking on a pile causes the pile to open and to display each of its included pages. Note that the structure is recursive: a pile can contain a set of pages and next-level piles which in turn, can be opened again.

Displaying an amount of information at the meta level, of interest to the user, is a critical issue. The options range between a "maximal" approach in which each page of the underlying document is represented by a thumbnail at the meta level when the document is hit anywhere, to a "minimal" approach in which only those pages in the document that were actually hit are represented. Both options are unsatisfactory: the maximal option is essentially that what the present generation of document retrieval systems offers. Thus, if the document has a volume of say 400 pages, then the user is forced to sift trough this volume of thumb nails to locate the information of interest. On the other hand, the minimal approach reveals only the immediate pages containing the hits which is too little. Normally, the immediate context around the hit is of importance to gain the right amount of information. The display rules in the context base derive this information from the intra-document data and knowledge of the hits. The hit-list of components, produced by a user-query and returned by the document retrieval system, is fed to the display query-form of the context base. The result is packaged and sent to the client for display by the browser.

Figure 4 depicts the parse tree of a document. This information is kept, as we mentioned, in the `intra` relation. For reasons of brevity the following description is a simplification of the operation of the real system. We have omitted many details that are not essential to the understanding of the principles involved.

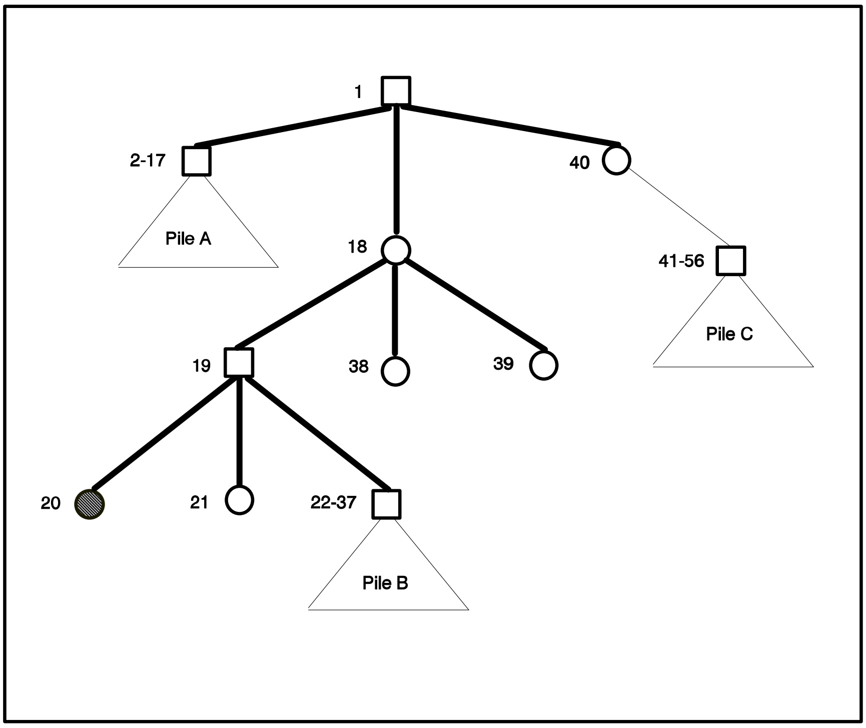

Fig. 4. Parse tree of a document

Each of the nodes has its own id (tag) and represents a document component.

As described in sec. 3, Nodes are typed, types being "paragraph," "section," "table," "figure," etc. A subset of the types is declared to be *stop_nodes*. A stop node, like stops in a zoom-lens, defines the next level of expansion or contraction of the information. Assume that the only stop type is "section" (a sub-section is also of type "section"). Thus, expanding a section reveals all of the components directly in this section and the headers of the next-level sections. In the Figure 4 the squares denote stop nodes. In the example {1, 2, 19, 22, 41} are stop nodes and the other nodes are non-stop nodes. Hits are depicted by a filled node. In this example node 20 is hit. Note that, in general, hits can occur anywhere in the tree. A hit to an interior node means that the header title of the corresponding component was hit.

The *scope* of a stop node N is the set of all non-stop nodes $\{N_1, N_2, \ldots\}$ such that,

1. there exists a path from each $N_i, i = 1, 2, \ldots$ to N, and,
2. the path does not include another stop-node $S, S \neq N$.

In our example, the scope of stop node 1 is {18, 38, 39, 40}.

An *expanded scope* is a scope containing node(s) that have been hit. In our example the scope of stop-node 19 is expanded and contains {19, 20, 21}. Note that the stop-node itself is included in the expansion.

A *pile* is a non-expanded scope. In the example we have included three piles. The root of each of these is labeled with the range of its id's. The root id of Pile A is 2 and its highest id is 17.

The *essential nodes* are the nodes in the expanded scopes and the nodes along the paths from roots of expanded scopes (subtrees) to the root of the tree. In our example the essential nodes are {1, 18, 19, 20, 21}.

Given these definitions we can now declare the display rules. For a given document the information that is displayed comprises of:

1. Essential nodes.
2. Direct siblings of essential nodes.

In our example piles included are A and B. The thick edges in the tree denote the subtree (nodes and piles) packaged for display. The second rule causes the inclusion of {38, 39, 40} and piles A and B.

4.3 Implementation and Performance Issues

It is important to distinguish between *document-invariant* and *hit-dependent* information. The parse-tree, sub-tree sizes and ranges as well as node and scope information are invariant. They can be derived and stored once, for the lifetime of the document. On the other hand, scope expansions, the determination of essential nodes and the final packaging for display are hit-dependent: this information can only be derived once the hit nodes become known.

In our initial (and in retrospect, naive) attempt to implement the display method we did not recognize this distinction and proceeded in a purely declarative fashion, described by the following program, simplified for expository purposes.

```
scope(X,Y) <-
        intra(X,_),
        stop_node(X),
        path(Y,X),
        ¬stop_node(Y).


expand(Z) <-
        hit(Y),
        scope(X,Y),
        scope(X,Z).


pile(X) <-
        scope(X,_),
        ¬expand(X).

essential(Z) <-
        path(Z,0),
        expand(Y),
        path(Y,Z).


display(X) <- essential(X).
display(X) <-
        essential(Y),
        intra(X,Y),
        pile(X).
```

The query `?display(X)` produces the requisite information for the package but at a heavy cost since the document-invariant data are computed over and again. A cost-effective solution to the problem combines the declarative with a procedural method: the intermediate results, derived by each of the rules in the program above, are stored by updating auxiliary relations in the context base. For example, the result of the `scope` derivation is stored as,

```
store_scope <-
        scope(X,Y),
        +scope_rel(X,Y).
```

Where `scope_rel(X,Y)` is an extension to the database to store this intermediate result. The `expand` rule uses its result for its own derivation and each subsequent result is produced, in sequence, by a derivation from the intermediate data produced at the previous step. Note that within each step the order of derivation is still undefined and immaterial. The performance improvement over the naive method is more than an order of magnitude: for a document on the order of 1500 nodes the naive method requires approx. 360 seconds while the declarative/procedural method described here requires approx. 10 seconds.

4.4 Inter-Document Knowledge

Up to this point we have discussed the logical structure of a single document and its uses in the abstraction of the informational content of the document. Other useful knowledge can be found and utilized when the relationships among different documents are considered as well. The upper part of the view in Figure 3 shows how the result of a user query is geographically distributed: the documents in the hit list are superimposed on a US map and are clustered by their site of origin. The user observes for example, that a certain topic which was the intention of his/her query, generated a large volume of documentation at Argonne National Laboratory, Argonne, Illinois, without having to descend to the actual documentation level to reach this conclusion.

To offer the information in this form, documents are linked by their site of origin and this information is kept in the context base. The user can move from the geographical ("where") view to e.g., a temporal view where the result of his query is superimposed on a time line so that he/she can conclude *when* a certain topic was of interest. From each of these views the user can move to each of the individual documents as described in the previous sections. The user can thus explore the information from different angles and rapidly form an opinion.

We have assumed that the "where" and "when" views of information are generic and of use to a wide user population. These views are thus included as a part of the present $\mathcal{IQ}$ system. The templates (maps, time lines) required to display them are permanently kept at the client side and query results are superimposed so that only the essential information is sent from the server. Other views of the information may be user-specific or specific to certain sub-populations of the users. For example, a user may be interested in all the documentation of person(s) with whom he/she collaborated in the past on a specific topic (a "whom" view). To implement such views, the user needs the capability to declare a set of rules, extending the context base schema for this purpose, as well as the necessary client templates. This $\mathcal{IQ}$ capability will be the topic of research of the next prototype of the system. We believe that the declarative nature of deductive database technology makes it very suitable for supporting this extension requirement.

5 Conclusion

In this paper we have described the prototype of a system, intended to enhance the user benefit of information retrieval from existing systems. The system allows for the continuous utilization of existing information retrieval systems albeit, at the expense of reengineering the content of these systems, as described in Sec. 3. There exists therefore a trade-off between the initial investment required for document preparation and the continuous benefit of the added context knowledge produced by the process. Over time, as the added cost of document preparation decreases due to the introduction of automatic tagging methods, it is clear that the approach advocated here is advantageous compared with current practices.

From a systems' perspective, $\mathcal{IQ}$ is a form of middle ware, interspersed between the underlying IR systems and the user-clients. It is not geared towards any particular method and can thus be used simultaneously with a multitude of different retrieval technologies. The economy of implementation is further enhanced by its integration with standard web tools for the distribution and visualization of the results, such as JAVA. The system is still in its first phase of development and it is too early to draw definitive conclusions regarding the use of Deductive Database technology as the implementation tool. At this stage we have demonstrated the feasibility of the concept of an Information Lens. The objectives ahead include the extension of these ideas to include inter-document knowledge as previously discussed and, most importantly, the real test will come when the $\mathcal{IQ}$ system is applied to large volumes of prepared documentation. Performance and acceptable response times for user-interaction are key to the success and still remain to be shown. Our experience so far suggests that the right balance between the procedural and declarative components is critical to achieving an acceptable performance; the use of $\mathcal{LDL}$++ allows for a great deal of flexibility in this respect.

DD technology has been developed in the context of structured, relational data. This application area demonstrates that there is a significant degree of structure in textual data which lends itself to treatment by the traditional database methods, which hitherto has not been addressed by the IR research community. We expect and hope therefore that this paper will contribute towards the opening of a "new frontier" in database research that will look into the combined use of structured and non-structured data.

Lastly, the subject of interacting with information via WWW will play an increasing role in this research area. The difficulties in describing these forms of interaction using the traditional text-based medium, such as this paper, have become painfully obvious. The authors would like, therefore, to use this opportunity to exhort the community to regard the Web and all of the rich information exchange opportunities it affords, not only as a subject of research but also as a medium and tool for the publication of the results and the organization of future meetings in this area.

References

[AOTZ] N. Arni, K. Ong, S. Tsur, and C. Zaniolo. $\mathcal{LDL}$++ : A second-generation deductive database system. Submitted for publication.

[Cro95] W. Bruce Croft. What Do People Want from Information Retrieval? (The Top 10 Research Issues for Companies that Use and Sell IR systems). *D-Lib Magazine*, November 1995.

[EXC95] http://www.excalib.com, 1995. Web Page, Excalibur Technologies, Inc.

[GM95] J. Gossling and H. McGilton. The JAVA Language: A White Paper. http://java.sun.com/whitepaper/java-whitepaper-1.html, 1995.

[HAR95] The harvest information discovery system. http://harvest.cs.colorado.edu, 1995. Web Page, Harvest Project, University of Colorado.

[PLS95] http://www.pls.com, 1995. Web Page, Personal Library Software Inc.

[Smi92] Joan M. Smith. *SGML and Related Standards Document and Processing Languages.* Ellis Horwood Limited, 1992.

[WAI95] Power tools for online publishing. http://www.wais.com, 1995. Web Page, WAIS Inc.

This article was processed using the LaTeX macro package with LLNCS style