

Boolean Query Translation for Brokerage on the Web

Boris Chidlovskii, Uwe M. Borghoff, Pierre-Yves Chevalier

Xerox Research Centre Europe, Grenoble Laboratory

6, Chemin de Maupertuis. F-38240 Meylan, France

E-mail: <surname>@xrce.xerox.com

Keywords : Web, heterogeneity, boolean query, query translation.

Abstract

Access to Web-based data repositories, including the sources available through cgi-search forms, creates a new dimension in distributed federated data processing. The heterogeneity of Web repositories poses new problems, one of which is the translation of Boolean queries between different repositories.

This paper highlights our understanding of Boolean query translation for brokerage on the Web that supports access to the heterogeneous data repositories. It introduces fundamental concepts used in the Knowledge Broker system currently under development in our center. We extend the model of query translation proposed in [Chang et al. 1996, Chang and Garcia-Molina 1997] and apply it to real Web repositories invoked for query brokerage.

Introduction

Distributed information retrieval and gathering on the Web relies upon the request brokerage and cooperation with multiple search services. Due to the heterogeneity of the Web, the services usually support query languages with various facilities for formulating a query. As the result, the user in a broker system is assumed to be provided with a front-end query language which usually unifies the various search facilities and hides the heterogeneity of the Web search services.

The front-end does not manage query and data internally, instead it relies upon external services to retrieve the information the user is asking for. To do this, the front-end must be able to translate user queries (written in the front-end unified language) to native languages supported by the Web information services.

If a user query contains operators not supported by a native language, the query translation on the first step leads to approximation of the original query with a query fitting into the native language. To guarantee a correct response to the user query, the approximated query subsumes the original one. As a result, the Web service response is the superset of data asked for by the original query. To eliminate the irrelevant data, the front-end performs post-filtering

as the second step of the query processing. As a primitive query subsumption might result in the inefficient utilization of the network resources, the query translation should be optimal.

Query Translation Problem

For years, the query translation was an in-core interoperability problem in heterogeneous databases sharing common databases but having different data manipulation languages [Florescu et al. 1995, Raschid et al. 1994]. The languages were assumed to have the same expressive language and the main problems were primarily in the integration of database schemes and serializability of updates. The appearance of the Web and new paradigms of seeing the data on the Web as a huge distributed resource of information, has changed and extended the domain of query translation.

Every Web information service holds a search engine as a tool to provide fast retrieval of data. What is common for different commercial and public/shareware search engines (Verity, Digital, Fulcrum, Excite, OpenTex, Glimpse, Swish, etc.) is a simple one-keyword query. The differences between the engines come to light when preparing a complex query. Most engines support Boolean operations **AND**, **OR**, **NOT** (=AND NOT) in full or at least reduced form. Binary proximity word operations like **W(n)** and **Near(n)** require two words to appear in a document in a given order or within a given distance **n** (usually, in words). Unary word operations usually include *phrases* and *stemming*. Finally, the *fielded search* predicates **CONTAINS** and **EQUALS** are used to restrict the search to some document fields, such as **Title**, **Author**, **Abstract** etc.

A fundamental case of query translation between search engines was studied in [Chang et al.1996, Chang and Garcia-Molina 1997]. It gives a theoretical underground and provides the efficient algorithm for optimal translation under the basic assumption that both front-end and native languages provide the Boolean operators **AND**, **OR** and **NOT** for the query formulation. The languages may however differ in sup-

porting the proximity operations **W(n)** and **Near(n)**, fielded search predicates **CONTAINS** and **EQUALS** etc.

For each operator or predicate that is supported differently in the two languages, a rewriting step is required to subsume the operator by operators available in the native language. As a result, the procedure of the query translation is as follows. First, a user query is transformed into the disjunctive normal form (DNF) which was proven to preserve the “minimality” property [Chang et al. 1996]. The predicate rewriting step then replaces the unsupported predicates with their subsumptions. Finally, the subsuming query is sent to the Web server. Once the response is received from the server, the post-filtering is used to discard the irrelevant data.

Example 1. Consider the query

$Q_1 = \text{Title CONTAINS fault W(2) tolerance}$

which should retrieve all documents with two words **fault** and **tolerance** in the title and with at most one word in between. If a native query language supports the Boolean operator **AND** and does not support the operator **W(n)**, we should subsume **W(2)** with **AND**. Therefore, the subsuming query is $Q^s = \text{Title CONTAINS fault AND tolerance}$. After the data has been retrieved from the service server, the post-filtering is called to discard those documents which do not meet the **W(2)** condition.

Query translation on the Web

The query model for query translation between two search engines [Chang et al. 1996] assumes that the Boolean operators **AND**, **OR**, **NOT** are supported by the native language. This results in guaranteeing the *one-query subsumption*, when *any* user query may be subsumed with one native query. Unfortunately, on the Web this assumption is not true for many information services.

Any Web search service is connected to an underlying search engine through, for example, a cgi-script. However, the user deals with the *Web page query language* rather than with the underlying search engine. The expressive power of the Web page query language can obviously not be higher than that of the underlying search engine. Nevertheless, even in the presence of a powerful search engine the Web page designer is often driven by the application needs which may result in a Web page query language which is far different from the underlying query language.

We can make the general statement that a Web page query language has an *operational limitation* if it fails to support all Boolean operators **AND**, **OR** and **NOT** or supports them in a non-standard way. In some cases, the basic query capacity is limited to just a one-word query. For example, services for searching FTP sites like Archie [Emtage and Deutsch 1992]

allow only one-word (regular expression) in a query. A number of Web services developed afterwards their own search engines for searching non-traditional data such as sequence or genome databases (e.g., FASTA-SWAP Pattern database <http://dot.imgen.bot.tmc.edu:9331>), most of which accept only one-word queries.

As an example of non-standard processing of the Boolean operators, we can consider the Yahoo search engine (<http://www.yahoo.com>). The objective of the Yahoo server is a wider search and better ranking of the information retrieved rather than reducing the amount of data. The order of keywords in the query serves for ranking the data. Then, ‘+’ appears in front of a word if it must appear in the response ‘-’ in front of a word is used to exclude documents containing the word. The standard Boolean query ‘**A AND B**’ therefore looks in Yahoo as ‘+**A +B**’, and the query ‘**A OR B**’ is ‘**A B**’, but there is no direct translation for the query ‘**A AND B OR C**’. In other words, Yahoo has an operational limitation, i.e., it supports either {**AND**, **NOT**} or {**OR**, **NOT**} Boolean operator sets.

To conclude our examples of operational limitations, some Web sites despite having a complete search engine in use, support only **AND** (or **OR**) operators to force the user to be more (less) selective during the search (as an example, ACM Digital Library at <http://www.acm.com>).

In this paper, we describe a solution to the query translation problem adopted in the Knowledge Broker [Borghoff et al. 1996, Chidlovskii et al. 1997]. We assume one-word query as a basic facility of a native language with Boolean operators being optional. We then study the cases of operational limitation and the possibility of one-query subsumption. For the cases when one-query subsumption is impossible, we show how the minimal number of subqueries can be used to subsume the original query.

The remainder of the paper is organized as follows. First, we consider the query translation architecture and the front-end query model adopted in the Knowledge Broker system. We then discuss the subsumption process for the case of operational limitation in a native language and the predicate rewriting rules. We also demonstrate the subsumption process by multiple examples. Finally, we conclude with some remarks on future work.

Query Translation Architecture

The architecture of the query translation in the Knowledge Broker system is close to that adopted in [Chang et al. 1996].

Figure 1 shows all steps of the query processing from submitting to returning the results. First, the user query is parsed and an internal tree-like presen-

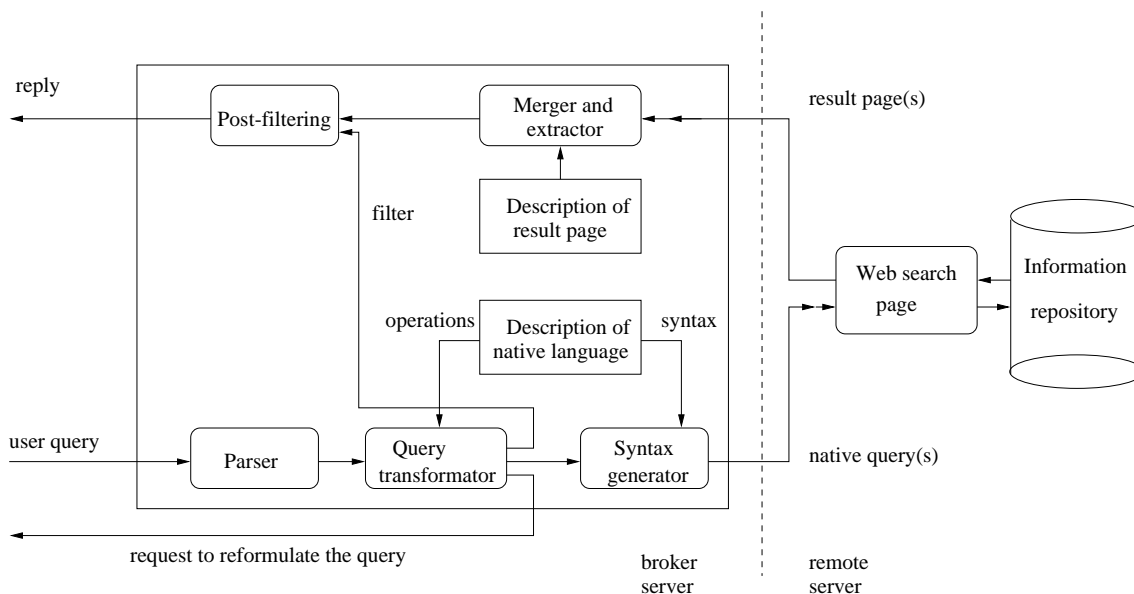


Figure 1: Query translation architecture.

tation is created. The query translator processes the query tree on the basis of operators supported in the native language to generate a subsuming query tree. Concurrently, the filter function for the post-filtering is generated. If the one-query subsumption is impossible, several subqueries are created. The subsuming query tree(s) are then put into the syntax of the native language and sent out through the net. After the raw html file(s) have been received and document items extracted from them, the post-filtering uses the filter function to discard the irrelevant items.

User query language

The front-end query model in the Knowledge Broker system includes a number of *basic predicates* which may be used to formulate a query. Below we group the basic predicates according to their semantics:

1. Boolean operators **AND**, **OR**, **NOT** (= **AND NOT**) are used to compose a complex query. The predicate **AND** indicates that both terms should be verified in a retrieved document while the predicate **OR** indicates that at least one of the them should be verified. The operator **NOT** is binary in the majority of IR systems and means that any retrieved document should satisfy the first term and not the second.
2. Binary word predicates between two word terms **A** and **B**:
 - query **A W(n) B** requires the term **A** to appear in the text within **n** words before **B**.
 - query **A Near(n) B** requires the terms **A** and **B** to appear in the text within **n** words, in any order.

3. Unary word predicates:
 - *phrase* is a quoted string (like '**information retrieval**') which is supposed to appear in the document.
 - **A*(stemming)** matches any words with the stem **A**. For example, **system*** matches **system**, **systems**, **systematic** etc.

4. A set of document *fields* (such as **Author**, **Title**, **Abstract** etc.) can be searched by using the field predicates **CONTAINS** and **EQUALS**. When using **EQUALS** to search a field, one or several phrases connected by only the Boolean operator **OR** are allowed. Instead, in the **CONTAINS** predicate any binary and unary predicates listed in the previous items may be used.

In addition, a special field **Date** can be searched with the predicates **BEFORE** and **AFTER** to retrieve documents dated before or after a given date.

Figure 2 gives the abstract syntax of the front-end query language in the Knowledge Broker system.

The internal presentation of a user query is a binary tree where an internal node contains a basic predicate and a tree leaf contains a word term, phrase or field name.

Example 2. Consider the query

$$Q_2 = (\text{Title CONTAINS distributed W(2) system*}) \text{ AND } (\text{Author EQUALS Lampert}).$$

The internal query presentation is the binary tree shown in Figure 3. Field predicates **CONTAINS** and **EQUALS** are considered as binary non-symmetric operators with the left daughter containing the field

Query	::=	Predicate (AND OR NOT) Query
		Predicate
Predicate	::=	Field CONTAINS C-Expression
		Field EQUALS E-Expression
		Date (BEFORE AFTER) DateTerm
E-Expression	::=	E-Expression OR Phrase
		Phrase
C-Expression	::=	C-Expression (AND OR NOT) C-Expression
		ProximityExp
		WordTerm
		Phrase
ProximityExp	::=	WordTerm W(n) WordTerm
		WordTerm Near(n) WordTerm
Phrase	::=	' (Word)+ '
WordTerm	::=	Word
		Word* //stemming

Figure 2: Abstract syntax of the front-end query language.

name and the right daughter containing the query term.

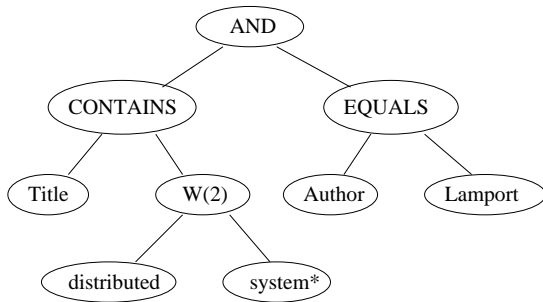


Figure 3: Query tree example.

Query Subsumption

Translation of the user query Q into a native query may often be done in different ways. Our goal in the Knowledge Broker system is to construct a native query which returns a minimal number of irrelevant “extra” data. We say that such a native query Q_s *minimally subsumes* the user query.

Like [Chang et al. 1996], we use the disjunctive normal form (DNF) of a query to preserve the “minimality” of translation. The DNF of the user query is a disjunction of the form $Q^{dnf} = C_1 \vee C_2 \vee \dots \vee C_k$, where each term C_i is a conjunction of predicates of the form $C_i = \tilde{P}_{i1} \wedge \tilde{P}_{i2} \wedge \dots \wedge \tilde{P}_{in}$. Each predicate \tilde{P}_{ij} is either a basic predicate P_{ij} or its negation $\neg P_{ij}$.

Example 3. The DNF of the query

$Q_3 = \text{Title CONTAINS Web}$
 $\text{NOT Usenet OR Internet}$

has two conjunctions $Q_3^{dnf} = \tilde{P}_{11} \wedge \tilde{P}_{12} \vee \tilde{P}_{21}$ with two basic predicates $P_{11} = \text{Title CONTAINS Web}$ and

$P_{12} = \text{Title CONTAINS Usenet}$ in the first conjunction and one basic predicate $P_{21} = \text{Title CONTAINS Web}$ in the second one. Note the predicate P_{12} is negated.

To generate the minimal subsumption Q_s for query Q , all unsupported basic predicates in Q^{dnf} should be rewritten by using the supported predicates. Each unsupported predicate P should have *positive* and *negative subsumptions* P_s^+ and P_s^- such that $\langle P_s^- \rangle \subset \langle P \rangle \subset \langle P_s^+ \rangle$, where $\langle P \rangle$ is the set of documents returned by a query P . That is, the subsumptions P_s^+ and P_s^- return a superset and subset of documents required by P .

During the predicate rewriting, each entry of P in the user query is replaced with the positive subsumption P_s^+ since $\langle P \rangle \subset \langle P_s^+ \rangle$, and each entry of its negation $\neg P$ is replaced with P_s^- since $\langle \neg P \rangle \subset \langle P_s^- \rangle$. Besides the positive and negative subsumptions, a predicate P may have a *neutral* subsumption $P_s^=$ supported by the native language, such that $\langle P \rangle = \langle P_s^= \rangle$. All subsumptions for the basic predicates are discussed in the following section.

Filtering function. The filtering function which is necessary for post-filtering is generated during the query translation process. The filtering function looks like any query and is initially equivalent to the user query tree. The nodes for supported predicates and predicates having a neutral subsumption are removed and the tree is obtained by contacting the tree after the removal of the nodes. The nodes left in the filter tree contain the basic predicates subsumed during the translation.

Basic predicate	Native language	Type	Subsumption
A AND B	AND not supported	positive	A or B
A OR B	OR not supported	positive	sub-queries A and B
A NOT B	NOT not supported	positive	A
A Near(n) B	OR, W(n) $\forall n$ Near(m), $m > n$ AND	neutral positive positive	(A W(n) B) OR (B W(n) A) A Near(m) B A AND B
A W(n) B	W(m), $m > n$ Near(m), $m \geq n$ AND	positive positive positive	A W(m) B A Near(m) B A AND B
'A ₁ A ₂ ... A _k ' (phrase)	W(0) W(m), $m > 0$, Near(0) Near(m), $m > 0$ AND	neutral positive positive positive positive	A ₁ W(0) A ₂ W(0) ... W(0) A _k A ₁ W(m) A ₂ W(m) ... W(m) A _k A ₁ Near(0) ... Near(0) A _k A ₁ Near(m) ... Near(m) A _k A ₁ AND A ₂ AND ... AND A _k
A*(stemming)	OR	neutral	A ₁ OR ... OR A _p
attr EQUALS A	CONTAINS	positive	attr CONTAINS A
attr CONTAINS A	full text search	positive	A

Table 1: Table of basic predicate subsumptions.

Subsumptions for Boolean operators

We have seen that the native Web page language can support an incomplete set of Boolean operators when at least one of the operators **AND**, **OR** or **NOT** is missing. Moreover, if the Boolean operator set is incomplete, the one-query subsumption does not work, i.e., some user queries cannot be subsumed with one native query.

Basically, Boolean operators need only positive subsumptions due to the DNF query presentation where the need for a negative subsumption never arises. Positive subsumptions for operators **AND** and **NOT** are rather easy and may be subsumed by one of the two terms. For **A AND B**, it may be either **A** or **B** since $(A \text{ AND } B) \subset A$ and $(A \text{ AND } B) \subset B$. For **A NOT B**, the first term **A** may subsume the operator since $(A \text{ NOT } B) \subset A$.

Missing **OR** gives more trouble than missing **AND** or **NOT**. Indeed, the set {**OR**} is the minimal set of Boolean operators necessary for the one-query subsumption. In other words, the presence of operator **OR** guarantees that any user query may be subsumed with one native query. If a native language does not support **OR**, subsumption with one native query can not be guaranteed for most queries. In such a case, the query should be split into the minimal number of sub-queries to be sent to the target server.

However, there does exist a group of queries containing **OR** which can be subsumed with one native query. To check this possibility, the query should be transformed into the conjunctive normal form (CNF). If the CNF contains a disjunction with one predicate P' only, the predicate can be used as a subsuming query. Moreover, this requirement for one-

query subsumption can be easily reformulated for the query DNF which is used for the internal presentation of queries in the Knowledge Broker. Namely, if all conjunctions C_i of the DNF contain one common positive predicate P'' , it can be used as a subsuming query.

All rewriting rules for the basic predicates including Boolean operators are collected in Table 1. As one predicate may have several rewriting rules (for different native languages), the rules are grouped by predicates given in the first column of the table. The second column shows different combinations of supported predicates. The third and fourth columns show the subsumption type and the subsumption itself. As negative subsumptions are easily derived from positive ones, Table 1 reports only neutral and positive subsumptions.

Word predicates

Word predicates compose the largest group of rewriting rules. Consider, for example, rules for the operator **Near(n)** when it is not supported by a native language. The operator can obviously be subsumed with **W(n)** or, if the later is not supported, with the **AND** operator. Table 1 also contains a special case of subsumption for **Near(n)**, when a native language supports **Near** only for some values of m (like in Altavista where **Near** is supported with the default value $m = 10$).

Table 1 proposes similar rewriting rules for the proximity operator **W(n)** and any word phrase which is equivalent to a sequence of words connected with the operator **W(0)**.

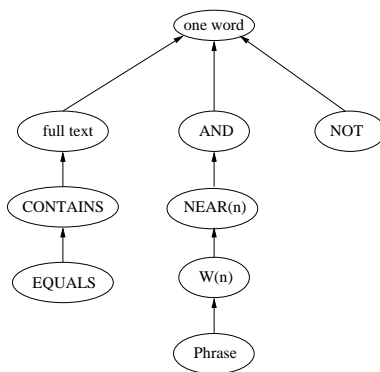


Figure 4: Predicate rewriting graph.

Field predicates

Before any Web service is included in the Knowledge Broker, we detect a list of queryable fields of the service, including the full-text search where available.

When subsuming a field predicate **EQUALS** or **CONTAINS**, we first check whether the field used in the predicate is queryable. If so, the positive subsumption for the predicate **EQUALS** uses the predicate **CONTAINS** instead (see Table 1). In contrast, **CONTAINS** has no positive subsumption with **EQUALS**¹. If no field search is possible but the full-text search is available, both predicates **CONTAINS** and **EQUAL** are rewritten as full-text search queries.

Note that the table provides no positive subsumption for any predicate (**BEFORE** and **AFTER**) against the field **Date**. Indeed, only if the native query language supports a similar field, can such a predicate appear in the subsuming query.

The final issue is when multiple predicate rewriting is necessary. For example, this happens during the translation of a query with predicate **Near(n)** to a native language supporting only one-word queries. In such a case, **Near(n)** is first rewritten with **AND** and the last is rewritten in a one-word query.

In Figure 4 we represent the predicate rewriting rules as a graph where each node corresponds to one predicate and each edge corresponds to one rewriting rule. One transition in the graph is one subsumption. Multiple rewriting corresponds with a chain of transitions in the graph. The predicate rewriting stops when all predicates in the query fit into the native language. Note that multiple rewriting does not concern queries with operator **OR** and the graph in Figure 4 contains no node for the operator.

Another basic predicate not included in the graph is the stemming as it is subsumed with the use of operator **OR**. In the real application of the Knowl-

¹The need of subsuming **CONTAINS** with **EQUALS** appears only if a Web page language supports **EQUALS** and not **CONTAINS**. It should be noted that this case is extremely rare on the Web.

edge Broker system, when the stemming is not supported remotely, it is actually solved locally by invoking the Xerox linguistic tools (available at <http://www.xrce.xerox.com/mltt>).

Query translation algorithm

The query translation procedure we report in Figure 5 deals with the two main situations occurring when a native language supports the Boolean predicate **OR**. If **OR** is supported, the query translation uses the query DNF and predicate rewriting rules. In the case when the predicate **OR** is not supported, the procedure tries to find a common predicate in all conjunctions of the DNF for one-query subsumption. If this fails, the procedure uses recursive calls to generate a sequence of sub-queries to subsume the original query. Concurrently, filter function(s) for the post-filtering are generated.

Predicate subsumption examples

In this subsection we give examples of the subsuming queries and filter functions for queries mentioned previously in the paper:

1. Query Q_1 when the native language supports one-word queries only:

Subsuming query

$$Q_1^s = \text{fault.}$$

Filter function

$$F(Q_1) = \text{Title CONTAINS fault} \\ \text{W(2) tolerance.}$$

2. Query Q_2 when predicates **CONTAINS** and **CONTAINS CONTAINS** are supported:

Subsuming query

$$Q_2^s = \text{Title CONTAINS distributed AND} \\ \text{AUTHOR CONTAINS Lamport.}$$

Procedure SUBSUME (query Q)
Input: query Q in its DNF: $Q = \bigvee_{i=1}^k C_i = \bigvee_{i=1}^k (\bigwedge_{j=1}^{p_i} \tilde{P}_{ij})$
Output: the minimal subsuming query(s) and the filter function.

begin
 Create the filter function as a copy of the query Q .
if OR is supported or Q contains no OR **then**
 for each basic predicate P_{ij} in the query, do the following:
 if predicate P is supported **then** remove it from the filter tree
 else // Subsume the predicate P_{ij}
 if exist the neutral subsumption **then**
 rewrite the predicate and remove it from the filter tree
 else if $\tilde{P}_{ij} = P_{ij}$ **then** replace P_{ij} with its positive subsumption
 else if $\tilde{P}_{ij} = \neg P_{ij}$ **then** replace P_{ij} with its negative subsumption
 return the subsuming query and the filter
else
 if exists a common term P_1 for all C_i such that $Q = P_1 \wedge Q'$ **then**
 return the subsuming query P_1 and filter Q'
 else for $i = 1, \dots, k$ call SUBSUME (C_i)
end

Figure 5: Query subsumption procedure.

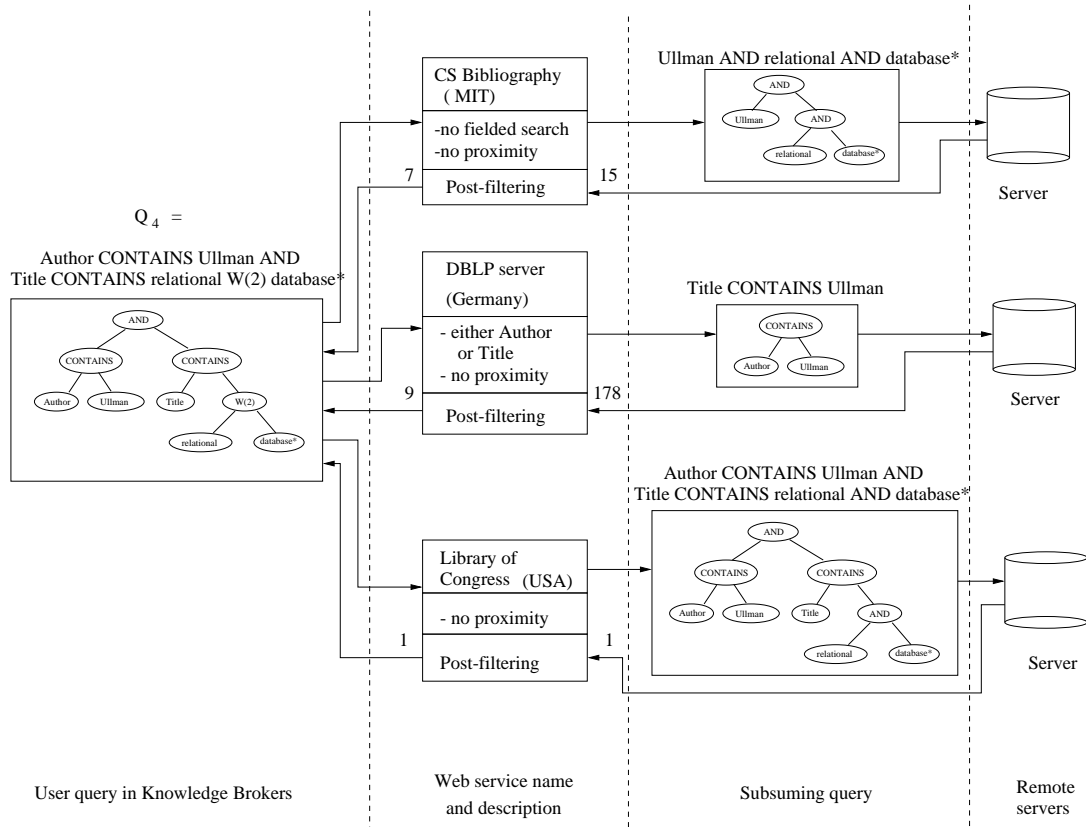


Figure 6: Query translation example for three Web servers.

Filter function

$$F(Q_2) = (\text{Title CONTAINS distributed} \\ \text{W(2) system* AND} \\ \text{Author EQUALS Lamport.})$$

3. Query Q_3 when one-word queries are only supported:

Two subsuming sub-queries

$$Q_{31}^s = \text{Web} \\ Q_{32}^s = \text{Internet.}$$

Filter function

$$F(Q_3) = \text{Title CONTAINS Web} \\ \text{NOT Usenet OR Internet.}$$

Finally, an example of the query translation for three real Web services is given in Figure 6. It shows the user query Q_4 , three Web services (Database and Logic Programming server (at <http://www.informatik.uni-trier.de/~ley/db/index.html>), the CS Bibliography at MIT (at <http://theory.lcs.mit.edu/~dmlones/hbp/>), and the Library of Congress (at <http://lcwebloc.gov/cgi-bin/zgate>), and describes the differences of their query languages from the front-end. For each service, an appropriate subsuming query is generated and sent to the service server. A number of documents before and after the post-filtering show the discrepancy between the subsumptions and the original query. For the Library of Congress, the subsumption concerns the proximity operator only and the post-filtering finds no irrelevant data. Instead, the subsumption for the DBLP service requires considerable modification of the original query. As a result, the DBLP server returns 178 documents of which only 9 pass the post-filtering step.

Conclusion

We have presented the Knowledge Broker approach to binary query translation on the Web. When the user query cannot directly fit into the native query language, it is substituted with the minimal subsuming query in order to minimize the network communication overhead and response time. We paid particular attention to cases where an information service does not support the universal set of Boolean operators and the query should be split into a number of sub-queries to search the service server.

Our future work will be on further analysis of the heterogeneity of the Web repositories and problems related to efficient query translation. We will study how the limitation on response sizes (typical

for many Web services) can influence the query translation quality and how to measure the accuracy of the returned responses for Web services where the verification of the response correctness is not always possible or needs more navigation (like Altavista).

References

- [Borghoff et al. 1996] Borghoff U. M., P.-Y. Chevalier and J. Willamowski. 1996. Adaptive Refinement of Search Patterns for Distributed Information Gathering, *Proc. Int'l Conf. EuroMedia'96/WEBTECH*.
- [Chang et al. 1996] Chang C.-C. K., H. Garcia-Molina and A. Paepcke. 1996. Boolean Query Mapping Across Heterogeneous Information Sources. *IEEE Transaction on Knowledge and Data Engineering* 8, no. 4.
- [Chang et al. 1996a] Chang C.-C. K., H. Garcia-Molina and A. Paepcke. 1996. Predicate Rewriting for Translation Boolean Queries in a Heterogeneous Information System, *Technical Report SIDL-WP-1996-0028*, Stanford University.
- [Chang and Garcia-Molina 1997] Chang C.-C. K. and H. Garcia-Molina. 1997. Evaluating the Cost of Boolean Query Mapping. *Proc. 2nd ACM Int'l Conf. Digital Library*.
- [Chidlovskii et al. 1997] Chidlovskii B., U. M. Borghoff and P.-Y. Chevalier. 1997. Toward Sophisticated Wrapping of Web-based Information Repositories, *Proc. Int'l RIAO'97 Conference*, Montreal, 123-135.
- [Emtage and Deutsch 1992] Emtage A. and P. Deutsch. 1992. Archie - an electronic directory service for the Internet. *Proc. USENIX Winter Conference*, 93-110.
- [Florescu et al. 1995] Florescu D., L. Raschid and P. Valduriez. 1995. Using Heterogeneous equivalences for Query Rewriting in Multidatabase Systems. *Proc. Cooperative Inform. Systems Conference*.
- [Papakonstantinou et al. 1995] Papakonstantinou Y., A. Gupta, H. Garcia-Molina and J. Ullman. 1995. A Query Transaction Scheme for Rapid Implementation of Wrappers. *Proc. DOOD'95 Conference*, Lect. Notes Comp. Science 1013, 161-186.
- [Raschid et al. 1994] Raschid L., Y. Chang and B. J. Dorr. 1994. Query Transformation Techniques for Interoperable Query Processing in Cooperative Information Systems. *Proc. Cooperative Inform. Systems Conference*.