Constraints-based Query Translation across Heterogeneous Sources for Distributed Information Retrieval

Lieming Huang, Ulrich Thiel, Matthias Hemmje, Erich J. Neuhold

GMD-IPSI, Dolivostr. 15, D-64293, Darmstadt, Germany

{lhuang, thiel, hemmje, neuhold}@darmstadt.gmd.de

ABSTRACT: In this paper we propose a method for translating queries across heterogeneous information sources with widely varying query forms. First, the query capabilities of specific sources are described while taking into account the information of various constraints. When translating a query from one source to another source, we also sufficiently consider the function and position restrictions of terms, term modifiers and logical operators among the controls in the user interfaces to the underlying sources, so we can utilize the query capabilities of the specific sources as much as possible. In addition, we put forward a two-phase query subsuming mechanism to compensate for the functional discrepancies between sources, in order to make a more accurate query translation.

1 INTRODUCTION

The number of queryable information sources on the Internet, such as search engines, search tools, and online repositories, is growing rapidly. The information explosion makes it difficult for even the most powerful search engine to index all web pages (there are innumerable local databases that can only be visited through their query interfaces on the Internet) and to index new and updated pages within a reasonable amount of time. Thus, sometimes it is difficult to satisfy users' information needs by only visiting one source. Efficient information integration systems (IISs, such as meta-search engines, information brokers, agent-based information systems, multi-databases, and so on) will undoubtedly facilitate users in acquiring information. Such systems integrate many distributed, heterogeneous information sources with quite differing query models and user interfaces. In this paper, we focus on automatic query translation across different sources. Considering the great diversity of heterogeneous Internet sources, it is certainly a significant task to design an efficient query mapping mechanism that can sufficiently coordinate the conflicts between sources.

Suppose that a user wants to search for information as follows: *Q*: ((Author is "Charlie Brown") *AND* (((“Information Integration" in All fields) *AND* (Title contains "query")) *OR* (("Metadata", "XML") in Abstract))), published during the period of 1995 to 1999.

Before discussing how this query can be translated into the formats supported by several concrete sources, we first briefly introduce the method we have used to describe the query interfaces and query capabilities of different information sources. In Fig. 1, we divide all the controls in the query construction user interface to a source into three groups: (1) classification selection controls, (2) result display controls, and (3) query input controls.

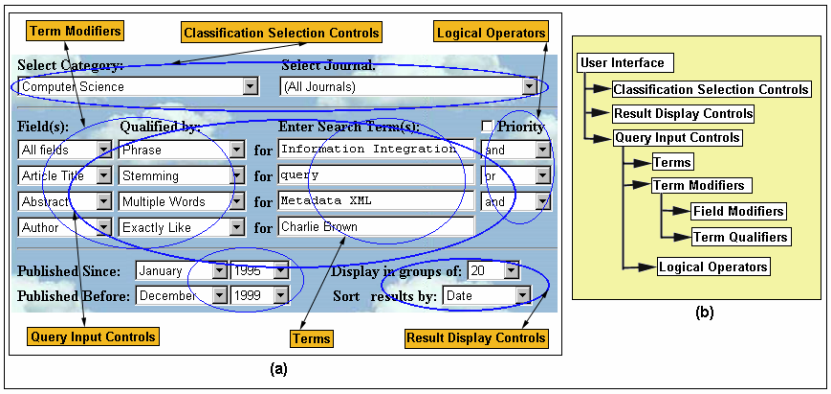

Fig. 1 Classifying the controls in a query form

A classification selection control is a component in the user interface to an information source which allows users to select one or more items in order to limit their information needs to certain domains, subjects, categories, etc.

A result display control can be used by users to control the formats, sizes or sorting methods of the query results. For example, Sorting Criteria = {<Relevance ranking>, <Author>, <Date>, etc.}; Grouping Size = {<10>, <20>, etc.}; Results Description = {<full>, <brief>, <URL>, etc.}.

A query input control is a component in the user interface to a source through which users can express their information needs (queries). Each query input control belongs to one of the three types: (1) TERMS. A "*term*" is the content keyed into an input box in the user interface to a source. This is different from the usual meaning of "term", because a term in our sense can be a single keyword, multiple words, a phrase, or a Boolean expression. In some cases, the input term may support wildcards, truncation, or stemming. It may be case sensitive, and might drop stop-words, hyphens, diacritics and special characters. (2) TERM MODIFIERS. A "*term modifier*" is a control in the user interface to a source that is used to limit the scope, the quality, or the form of a term. For example, we can divide all term modifiers into two groups: (i) field modifiers: {<Title>, <Full-Text>, <Review>, < Keywords>, etc.}, (ii) term qualifiers: {<Exactly Like>, <Multiple Words>, <Using Stem Expansion>, <Phrase>, etc.}. (3) LOGICAL OPERATORS. A "*logical operator*" is a control in the user interface to a source that is used to logically combine two terms to perform a search, the results of which are then evaluated for relevance. For example, there are four logical operators: {AND, OR, NOT, NEAR}.

Based on this classifying method, we can describe the query capabilities of various information sources, especially those with complex, powerful user interfaces, such as search engines with advanced query input forms, digital libraries, etc. In the following, we show how the example query *Q* is posed in two concrete query interfaces (Figures 2-3) from various sources, thus demonstrating the great diversity in the query models and user interfaces of heterogeneous sources, and illustrating the difficulties involved in translating a query from one source to another.

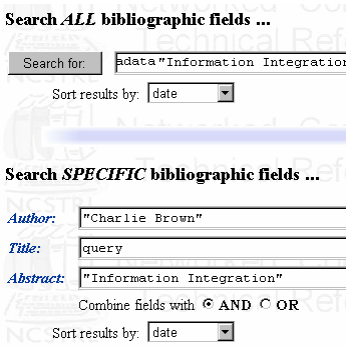

Fig. 2 NCSTRL

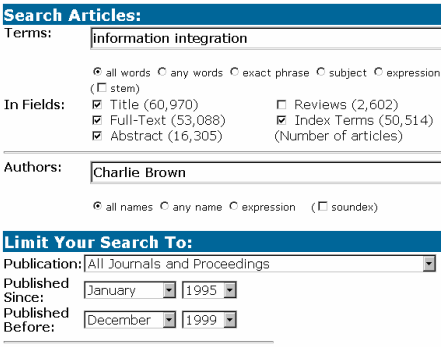

Fig.3 ACM-DL

In Fig. 2, two separate fill-out forms of the NCSTRL search engine are shown. The first form ("Search ALL bibliographic fields ...") contains only one input box and a result-sorting criteria pull-down menu. We can put all words and phrases into the input box. However, the field modifiers (Title, Abstract, Author, etc), term qualifiers (Exactly like, Using stem expansions, etc), and date range cannot be expressed in this form, and the search is likely to retrieve irrelevant results with respect to the original query. The second form ("Search SPECIFIC bibliographic fields ...") contains three input boxes (each input box is associated with a single field modifier: 'Author', 'Title', and 'Abstract'), two radio boxes as logical operators (AND, OR) used to combine the three input boxes, and also a result-sorting criteria pull-down menu. Compared with the first form, the second form can better support the query *Q*. Fig. 3 shows the query page of the ACM-DL with an input box for users to input keywords (it has several term qualifiers), five check boxes in which users can select field modifiers (title, full-text, abstract, review, article keywords), and one author input box.

From the above two figures, we can see that there is great diversity among the user interfaces and query models of various information sources, and that translating a query across sources is not trivial. An information integration system may integrate hundreds or even thousands of information sources. Depending on some source selection algorithms, the system can choose some sources that are relevant to a user query and then translate the user query into the formats supported by these sources. In the next section, we will discuss how such query translations are carried out.

2. CONSTRAINTS-BASED QUERY TRANSLATION

In this section, we will introduce the process of constraints-based query translation. Section 2.1 introduces how an original query can be disjunctivized into several conjunctive sub-queries. Section 2.2 discusses the translation from a conjunctive query into a single target query and how the common filters and the special filters are generated and how they work. Section 2.3 gives a detailed example to explain this translation method. Finally, section 2.4 briefly discusses the translation from an arbitrary query into several target query expressions.

2.1 Query Disjunctivizing

If a user query contains 'OR' logical operators, it can be transformed into several conjunctive sub-queries (the terms of each sub-query are combined by AND, NEAR, or NOT logical operators). For example, the query *Q* can be decomposed into 2 sub-queries: Q_1: ((Author is "Charlie Brown") *AND* ("Information Integration" in All fields) *AND* (Title contains "query") *AND* (Published during the period of 1995 to 1999)); and Q_2: ((Author is "Charlie Brown") *AND* (("Metadata", "XML") in Abstract) *AND* (Published during the period of 1995 to 1999)).

In order to explain our query translation method more clearly, we use figures to express the query capabilities of query forms. For example, query Q_1 can be described as Fig. 4 and the query capability of the second form (See Fig. 2) of the NCSTRL search engine can be described as Fig. 5.

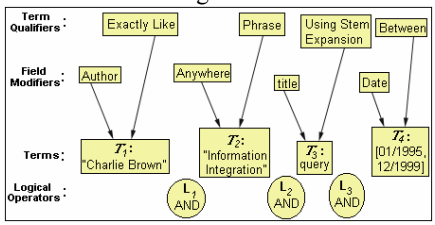

Fig. 4 Description of Q_1

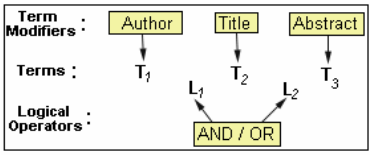

Fig. 5 the second form of NCSTRL

Because each field in Fig. 5 can only be limited by a specific term modifier, when we translate Q_1 into the formats supported by Fig. 5, the second term ("Information Integration" in All fields) in Q_1 can only be mapped into three concrete fields in Fig. 5: <Title>, <Abstract> and <Author>. So Q_1 can be disjunctivized into three sub-expressions: $Q_{1,1}$: (T_1) AND (T_2: Title) AND (T_3) AND (T_4); $Q_{1,2}$: (T_1) AND (T_2: Abstract) AND (T_3) AND (T_4); and $Q_{1,3}$: (T_1) AND (T_2: Author) AND (T_3) AND (T_4). Apparently, the third sub-query $Q_{1,3}$ contains the term (Authors contain "Information Integration") and this query will retrieve nothing.

A query and its disjunctivized sub-queries have the same effects, i.e. the retrieved results of these two situations are same. Sometimes an original

disjunctive query can be directly mapped into a target query without being transformed.

2.2 Translation of a conjunctive query into a single target query

Now we discuss the translation of generic query expressions. We suppose that the original query Q^o is a conjunctive query with m (m>0) terms and (m-1) logical operators (can be 'AND', 'NOT', or 'NEAR') and that the target query Q^t is a query with n (n>0) terms and (n-1) logical operators (See Fig. 6).

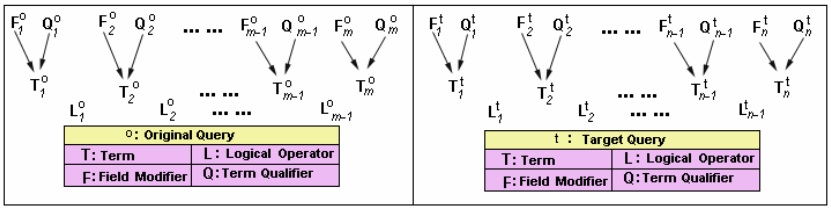

Fig. 6 the original query $\boldsymbol{Q^o}$ and the target query $\boldsymbol{Q^t}$

When the system translates Q^o into Q^t, one of the following three cases will occur. Fig. 7 illustrates these three cases. Now we discuss these three cases separately and at the same time introduce how the common filters (these kinds of filters occur frequently and most of them can be applied to refine the results, so we call them "Common") and the special filters (these kinds of filters occur not often and most of them cannot be applied to refine the results, so we call them "special") are generated and how they later will be used to post-process the raw results.

Case 1: In this case, each term in Q^o can be put into a certain term in Q^t, and the field modifier and the term qualifier of this term can also be supported by the corresponding term in Q^t. Furthermore, each logical operator in Q^o can also be supported in Q^t, and the logical value of the new query is equivalent to the original query if some terms exchange their positions. For example, (A AND B AND C NOT D) equals (B AND C AND A NOT D). We call this situation as "Perfect Match" because the results need not be post-processed. In the following we give a more detailed description of this case.

For each term T^o_i in Q^o, if the field modifier of this term is one (or a subset) of the field modifiers of the corresponding term T^t_j in Q^t, and the term qualifier of T^o_i is one (or a subset) of the term qualifiers of T^t_j, then we consider the following three situations (otherwise, this translation fails): (a) If T^t_j is empty and the logical operator L^t_{j-1} supports L^o_{i-1} (L^t_{j-1} can be set as L^o_{i-1}), then the system can put T^o_i into T^t_j, set the field modifier and term qualifier of T^t_j as the corresponding ones of T^o_i, and set L^t_{j-1} as L^o_{i-1}. (b) Otherwise, if the term qualifier of T^t_j supports <Multiple Words> (the term T^t_j can be several words, but not a phrase) and L^t_{j-1} equals L^o_{i-1}, then put T^o_i into T^t_j. (c) Otherwise, Q^o cannot be translated into Q^t, and this translation fails.

If all terms in Q^o satisfy situation (a) or situation (b), the translation is successful. Otherwise, this query will be transferred to the next stage (Case 2) of the query translation.

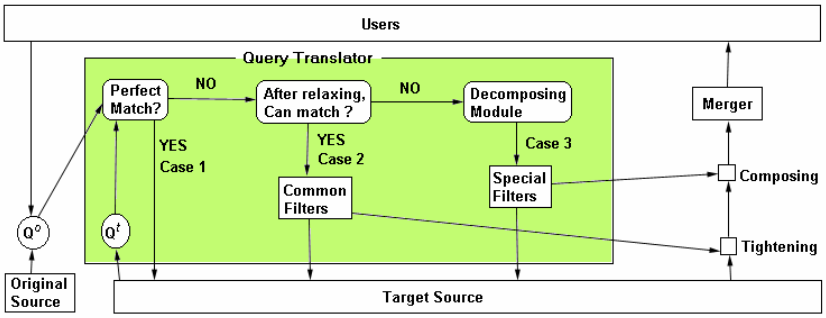

Fig. 7 Three cases of query translation

Case 2: Some field modifiers, term qualifiers or logical operators in Q^o cannot be supported by Q^t, but after relaxing them (i.e. broadening the scope of the limitation and therefore enabling that more results may be retrieved), for example, 'NEAR'→'AND', <Phrase> → <Multiple Words>, <Title> → <Abstract> → <Full Text>, etc., the newly-generated Q^o can be supported by Q^t. In this case, the IIS dispatches the relaxed query, and when the results come, the system then post-processes the results according to the previous relaxing information. For the relaxed field modifiers, term qualifiers and logical operators, the system will use some filters to record such information and later use them to refine the results in order to compensate for the relaxing of constraints. We call such filters "common filters" and call the result refining process as "Tightening" (See Fig. 7). Now we will discuss some common filters: (1)'NEAR'→'AND'. Many sources do not support 'NEAR' logical operators. The system uses "A AND B" to replace "A NEAR B" and generates a new common filter to record this information. After the results are retrieved, the system uses this filter to select those entries in which term A and term B are near each other (e.g. within 5 words). If term A and term B are in the 'Title' field, this post-processing is easy, but if they are in the 'Abstract' field or even in the 'Full-Text' field, the system will consider the cost of analyzing the content of the source file and then decide whether to post-process it. (2) 'Phrase'→'Multiple words'. For this case, the system chooses those results that contain the exact phrase. (3) 'NOT A'. Some sources do not support the 'NOT' logical operator. When translating the original query, the system discards the 'NOT' operator and its term and generates a new common filter to record this information. After the results come, the system uses this filter to remove the results containing the terms that the original query "NOTted".

Case 3: In this case, Q^o cannot be supported by Q^t even after relaxing some modifiers or logical operators. The system will break Q^o into several sub-queries, then translate and dispatch each sub-query separately. We use special filters to

record such decomposition information (See Fig. 7). When the corresponding results come, these "special filters" are employed to compose the results. However, in most cases, either because we cannot obtain relevant information from target sources or because post-processing will cost unreasonable CPU-time, it is impossible to post-process broken conjunctive expressions. For example, suppose that a four-term query is (A *AND* B *AND* C *AND* D) and the target query only supports two terms. Now we decompose the original expression into two sub-expressions (A *AND* B) and (C *AND* D). If the four terms are limited to the "Abstract" field or the "Full-Text" field of the publications, we cannot intersect the two result sets from (A *AND* B) and (C *AND* D) because we cannot check whether a term is in such fields. Even if we can get such information (e.g. by analyzing the PS, HTML, or PDF source file), such strenuous work is unnecessary. If the four terms are in the "title" field of the publications, it is possible to check if each entry from the two result sets contains these four terms. If the post-processing costs a lot of time, it is better to directly display the raw results to users.

When translating the original query into the target query, three steps (i.e. disjunctivizing, decomposing, relaxing) need to be accomplished, in which one or more of these steps may be skipped depending on the actual situation. When transferring the results from a source to users, three corresponding steps (i.e. tightening, composing, merging) will be done. The common filters record the relaxing information and later will be used to tighten the results. The special filters record the decomposing information and later will be used to compose the results.

2.3 An Example of Query Translation and Post-processing

Now, we discuss an example of how common filters and special filters are generated and later employed to post-process the results (See Fig. 8). Suppose there is a query six-term query Q^o: (A NEAR B AND C AND D AND E NOT F), term A and term B belong to the 'Abstract' field, term C belongs to the 'Author' field, term D belongs to the 'Title' field, term E belongs to the 'Full-Text' field and term F belongs to the 'Keywords' field. The target query Q^t can only support two terms and each term can only support the 'Author', 'Title', 'Abstract', and 'Keywords' field modifiers, and the 'AND' and 'OR' logical operators.

Because both term A and term B belong to the "Abstract" field, they can be put together into an input-box of the target source. Because Q^t cannot support the 'NEAR' logical operator, the system generates a new common filter *CF1: "NEAR->AND(A, B)"* and Q^o becomes ((A B) AND C AND D AND E NOT F). In this query expression, the first term is "A B" (because both the term A and term B belong to "Abstract" field, they can be put together in an input-box in the target source. For example, the term A is "XML" and the term B is "RDF", then the new term in the input-box can be regarded as two words: XML RDF, not a phrase "XML RDF".) and the second term is "C"

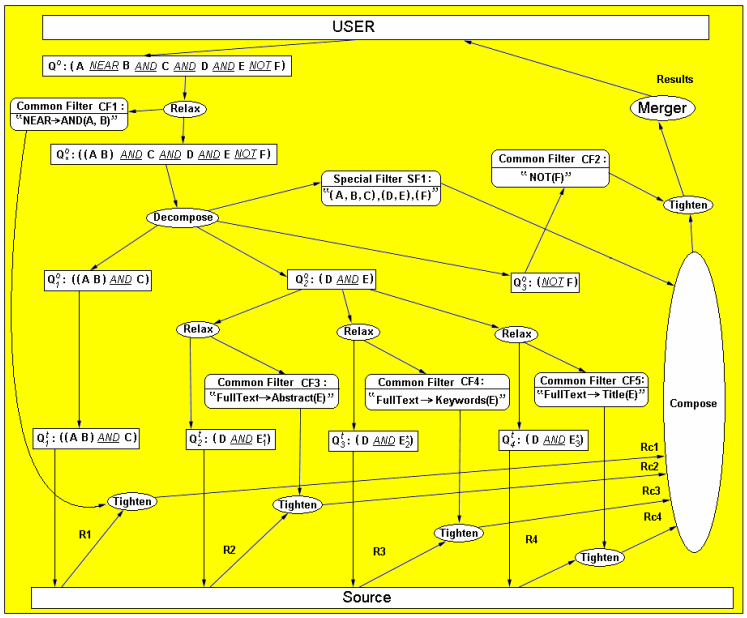

Fig. 8 An example query translation

Because the target source can only support two terms, when the system translates Q^o into Q^t, Q^o will be decomposed (See case 3 in Fig. 7) into three sub-queries: Q^o_1: ((A B) AND C), Q^o_2: (D AND E) and Q^o_3: (NOT F) and a new special filter *SF1: "(A, B, C), (D, E), (F)"* will be generated. Later this special filter will be employed to compose the three result sets of these three sub-queries.

Because the target source cannot support 'NOT' operator, so the sub-query Q^o_3: (NOT F) cannot be sent to the target source. Then a new common filter *CF2: "NOT(F)"* is generated and later this filter will be used to post-process the results. In the following, we will discuss how the system translates Q^o_1 and Q^o_2 into Q^t separately.

Q^o_1 becomes Q^t_1. In this Q^t_1, the first term is "A B" (because both the term A and term B belong to "Abstract" field, they can be put together in an input-box in the target source) and the second term is "C". When the system translates Q^o_2 into Q^t, because Q^t cannot support the 'Full-Text' field modifier, three new common filters *CF3: "FullText->Abstract(E)"* , *CF4: "FullText->Keywords(E)"* , and *CF5: "FullText->Title(E)"* are generated and Q^o_2 is transformed into Q^t_2: (D AND E^*_1), Q^t_3: (D AND E^*_2), and Q^t_4: (D AND E^*_3) repectively. In Q^t_2, the term E^*_1 belongs to the 'Abstract' field; In Q^t_3, the term E^*_2 belongs to the 'Keywords' field; And in Q^t_4, the term E^*_3 belongs to the 'Keywords' field.

After that, the system dispatches Q^t_1, Q^t_2, Q^t_3, and Q^t_4.

When the results of the query Q^t_1 return (R1), the system will use the common filter *CF1* (*"NEAR->AND(A, B)"*) to refine them as Rc1, i.e. choosing those

entries in which term A and term B are near each other within a number of (e.g., 3) words. When the results of the query Q^t_2 come (R2), the system will use the common filter *CF3* (*"FullText->Abstract(E)"*) to refine them as Rc2 (this filter will be skipped because the <Abstract> can almost be regarded as a subset of <FullText>). The common filters *CF4,* and *CF5* are the same as *CF3*. Then the system will use the special filter *SF1* (*"(A, B, C), (D, E),(F)"*) to compose the two results sets Rc1, Rc2, Rc3, and Rc4, i.e. intersecting the two result sets. Finally the system will use the common filter *CF2* (*"NOT(F)"*) to remove the entries that contain the term F in the 'keyword' field.

2.4 Translation from an arbitrary query into several target queries

In the above we have discussed this situation: the source query is a conjunctive query and the target source allows only one query expression. However, some sources provide more than one fill-out form, thus allowing more than one query expression, for example, NCSTRL supports two forms (See Fig. 2). Sometimes, the system may distribute several sub-queries from a source query into several target query expressions when translating the original query. Sometimes an original query can be directly mapped into the target query without being disjunctivized. Decomposing a query will increase the number of visits to remote sources and reduce efficiency. However, for most information sources, query decomposition is necessary.

3 RELATED WORK AND DISCUSSIONS

With the tremendous development of the Internet and the explosive growth of digital information, distributed information retrieval on the Internet becomes more and more important. The challenge of its difficulties and importance attracts the attention and efforts of many researchers, such as [1-7], to name a few. In the following, some related work will be discussed.

Papers [2, 3] apply user-defined mapping rules to subsume queries for translation between different sources. They describe some problems involved in predicate rewriting, such as the "contains" predicate and word patterns, the "equals" predicate and phrase patterns, proximity operators, etc. Compared with their work, we propose a more generic model for translating arbitrary queries supported by various sources. Our two-phase method for coping with query subsuming (relaxing and decomposing) and post-processing (tightening with common filters and composing with special filters) can well coordinate the great functional discrepancies among heterogeneous information sources. Many papers, such as [5, 6, 7], describe the query capabilities of sources and deal with the query translation problem. They discuss more on context-free, conjunctive queries and do not consider some special constraints, such as the limitations of term modifiers, logical operators and the order of terms. From Figures 2-3, we know there is great

diversity among sources. Sometimes even a very subtle difference will render the query translation impossible. Our paper sufficiently describes all kinds of constraints between the query models (as embodied in the user interfaces) of various sources, and therefore can utilize the functionality of each source to the fullest extent. Paper [1] uses Church-Rosser systems to characterize the query capabilities of information sources and uses "Attribute Preference Ordering" to realize query relaxing. Paper [4] proposes a scheme called "GenCompact" for generating capability-sensitive plans for queries on Internet sources. These two papers try to describe the query capabilities of sources and to translate queries across sources in a generic view. However, they do not consider some specific query constraints and do not provide a mechanism to post-process inexact results. We maintain that the heterogeneity of information sources inevitably renders the query mapping inaccurate, and that post-processing of results is necessary to make up for the inaccuracy.

The constraints-based query translation method proposed in this paper can be applied to all kinds of Internet information integration systems, such as digital libraries, meta-search engines (especially for specific-purpose), agent-based information providers, etc.

REFERENCES

[1] S. Adali and C. Bufi. A Flexible Architecture for Query Integration and Mapping. Proc. CoopIS'98, New York. Aug. 1998, pp. 341-353.

[2] B. Chidlovskii, U. M. Borghoff, and P.Y. Chevalier. Boolean Query Translation for Brokerage on the Web. Proc. 2nd Int'l Conf. EuroMedia/WEBTEC'98, Leicester, U.K. Jan. 1998, pp. 37-44.

[3] C. Chang, H. Garcia-Molina, and A. Paepcke. Predicate Rewriting for Translating Boolean Queries in a Heterogeneous Information System. ACM TOIS 17(1), Jan. 1999, pp. 1-39.

[4] H. Garcia-Molina, W. Labio, and R. Yerneni. Capability-Sensitive Query Processing on Internet Sources. Proc. ICDE' 99, Sydney, Australia, Mar. 1999.

[5] A. Levy, A. Rajaraman, and J. Ordille. Querying Heterogeneneous Information Sources Using Source Descriptions. Proc. 22nd VLDB. Bombay, India, 1996, pp. 251-262.

[6] V. Vassalos and Y. Papakonstantinou. Describing and Using Query Capabilities of Heterogeneous Sources. Proc. 23rd VLDB. Athens, Greece, Aug. 1997. pp 256-265.

[7] R. Yerneni, C. Li, H. Garcia-Molina, and J. Ullman. Computing Capabilities of Mediators. Proc. SIGMOD, Philadelphia, PA, Jun. 1999, pp. 443-454.