Querying Heterogeneous and Distributed Data Repositories using Ontologies

Alfredo Goñi[1] Eduardo Mena[2,*] Arantza Illarramendi[2]

[1]*Departamento de Informática e Ingeniería de Sistemas. Universidad de Zaragoza.*
María de Luna, 3. 50015 Zaragoza, Spain.
[2]*Facultad de Informática. Universidad del País Vasco (UPV/EHU)*
Apdo. 649, 20080 Donostia-San Sebastián. Spain.
http://siul02.si.ehu.es/˜jirgbdat

Abstract. The facility that allows users to query heterogeneous and distributed data repositories the majority of users. Moreover, that facility should hide all the intrinsic problems related to the context such as location, organization/structure, query language and semantics of the data in various repositories. In this paper a query processing strategy that focuses on information content is presented. For that, domain specific ontologies that capture the information content of data repositories are used. Those ontologies are described using a system based on Description Logics. It is also explained the different steps followed to provide incremental answers to user queries navigating across ontologies, using pre-defined semantic interontology relationships. Description Logics systems capabilities are exploited to optimize queries and guide the query processing.

1. Introduction

On the global information infrastructure, the great expansion of the communication networks has made available to the users a huge number of heterogeneous and autonomous data repositories. However, these repositories present different structures/organizations, query languages and data semantics, making very difficult for the users to access the data stored on them.

A possible solution to lighten the problem of lack of uniformity when dealing with the available repositories consists on defining new information retrieval techniques with a strategy that focuses on information content and semantics. We propose to describe the content of the repositories by ontologies [14]. We make available to users several domain specific ontologies from which they can construct queries. Ontologies give a concise and declarative description of semantic information independent of the underlying syntactic representation of the data. This information can be used to determine the relevance of the underlying data without actually accessing the data, thus enabling scalable query processing. Domain ontologies can also be used to capture new and different world views, thus enabling wider accessibility of data.

[*] This work was supported by a grant of the Basque Country Government.

In our proposal, ontologies are described using a system based on Description Logics (DL). Moreover, terms from those ontologies are linked to the underlying repositories through mapping informations that are expressed using the extended relational algebra. Reasoning mechanisms from DL are useful to perform query optimization and, in particular, semantic and caching optimization. DL systems are also appropriated to offer intensional answers to the users [12]. Mapping descriptions play a key role in encapsulating the heterogeneity due to different formats and organization of the data in the various repositories. They act as an intermediary language between the DL expressions and the query languages of the local repositories.

In the proposed framework of loosely-coupled ontologies, the query processor takes as input a user query expressed in Description Logics, using terms from a chosen ontology, and tries to find the answer in the underlying data repositories. When a full answer cannot be obtained because the data are spread over several repositories under different ontologies, then the system navigates other component ontologies (until the user is satisfied with the answer) of the global information infrastructure translating terms in the user query into the "language" of component ontologies. This determines the relevant data repositories under the component ontologies providing a solution for the resource discovery problem [20].

In the literature different approaches for query processing in global information systems can be found. However, they mainly can be classified into three groups: syntactic keyword-based approaches (e.g. [2, 15, 16]), operational approaches based on Mediators (e.g. [8]), and approaches that use semantically rich views (expressed using Object Oriented models or Knowledge Representation Systems) describing data sources, either a global view (e.g. Carnot project [9], Information Manifold project [17], Cooperative Information System [10]) or several views (e.g. [3, 22]).

Our approach is representative of the works that use several views; in particular we deal with several domain specific ontologies. Our contribution is that we tackle the vocabulary sharing problem by using translations of a user query constructed from concepts in one domain ontology to other domain ontologies in the system. We also consider the cases where there is loss of information and use well established metrics like *precision* and *recall* to measure this loss. However we do not study the problem of building the ontologies as they do in [21].

The main goal of this paper is the presentation of the defined query processing strategy that:

1. allows one querying heterogeneous and distributed data repositories in a smart way focusing on information content,

2. provides incremental answers in two different ways: extensional and intensional,

3. gives a measure of a loss of information for extensional answers when it corresponds, and

4. takes advantage of the capabilities provided by DL systems.

In the rest of the paper we present first the global framework, then the main steps followed when accessing data underlying an ontology. Next, the process of enriching the answer by managing other ontologies, and last we conclude with some details about the prototype that has been implemented.

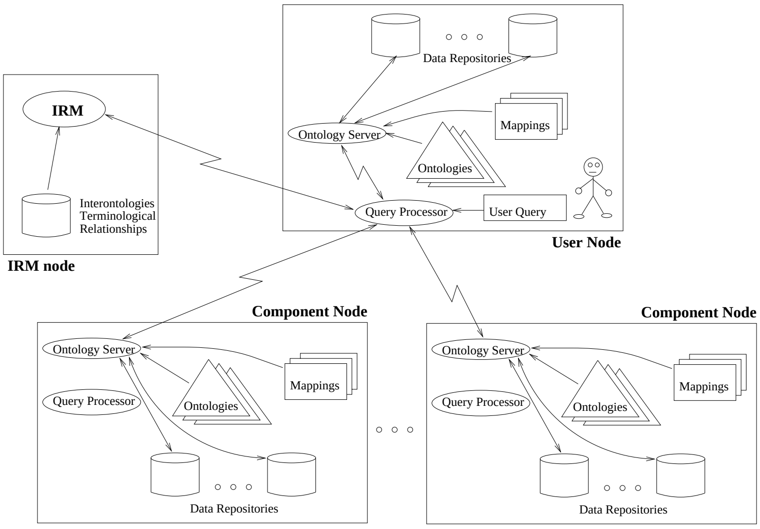

Figure 1: OBSERVER Global Architecture

2. Framework

The architecture of the OBSERVER[1] system that provides incremental answers to user formulated queries in a Global Information System has the following main components: Query Processor, Ontology Server, Interontology Relationships Manager (IRM) and the ontologies (see Figure 1). That architecture was explained in detail in [20] so we present it here briefly.

- *Query Processor.* It takes as input a user query expressed in DLs using terms from a chosen *user ontology.* The Query Processor communicates to the corresponding ontology Server which accesses the underlying data stored in the repositories under the user ontology. If the user is not satisfied with the answer, the Query Processor will translate the user query into the "language" of another (target) ontology by utilizing predefined terminological relationships (synonyms, hyponyms and hypernyms) between the user and the target ontology. The result is, in general, a list of translations or *plans* with an associated measure of the loss of information (that can be zero in the case of translating using only synonyms). The translation with less loss is chosen to access new data and the remaining plans are stored because they can be used later to upgrade the answer. Finally, the new answer (with or without loss of information) is correlated and presented to the user. This process is repeated until the answer satisfies the user.

- *Ontology Server.* The Ontology Server provides term definitions in the ontology and retrieves data underlying the ontology for the Query Processor. *Mappings* that link each term in an ontology with structures in data repositories are combined by the Ontology Server in order to access and retrieve data from the repositories with the help of wrappers. This addresses the *structure/format heterogeneity* problem. We describe in detail the access to underlying data sources in Section .

[1] OBSERVER (*Ontology Based System Enhanced with Relationships for Vocabulary hEterogeneity Resolution*) is our approach of using multiple pre-existing ontologies to access heterogeneous, distributed and independently developed data repositories [20].

- *Interontology Relationships Manager (IRM).* Terminological relationships relating the terms in various ontologies are represented in a declarative manner in an independent repository. This enables a solution to the *vocabulary sharing problem.*
- *Ontologies.* Each ontology is a set of terms of interest in a particular information domain; in our work, such terms are expressed using DLs. They are organized as a lattice and may be considered as semantically rich metadata capturing the information content of the underlying data repositories. These semantically rich descriptions can be used to query the Global Information System.

Description Logics

Systems based on DLs, also known as Terminological Systems, are descendants of *KL-ONE* [7] and allow us to define ontologies by using terminological descriptions. Some systems based on DLs are *CLASSIC* [6] (used in our prototype), *BACK* [24], *LOOM* [18] and *KRIS* [1]. The main features of the DL systems are described below:

- The language contains unary relations called *concepts* which represent classes of objects in the domain and binary relations called *roles* which describe relationships between objects. Concepts and roles are created via *terminological descriptions* built from preexisting concepts, roles and a set of operators (ALL, ATLEAST, ATMOST, etc.).

  Queries, that are also concept descriptions have the following format:

  [<projections>] for < concept-description >

  where < concept-description > defines the necessary and sufficient conditions that objects that form the answer verify. < Projections > contains the roles which values have to be projected.

- *Primitive and defined terms.* Terms (concepts and roles) are *primitive* if their descriptions specify only the necessary conditions and are *defined* if their descriptions specify both the necessary and sufficient conditions.
- *Subsumption of terms* allows to determinate whether a term is more general than another. The *subsumption* relationship is used by the DL system to maintain a classification hierarchy/lattice of terms (which is useful in dealing with large collections of definitions) and to *classify* new terms as well as queries. This classification mechanism allows the system to detect *incoherent* and *disjoint* descriptions.

  There exist two distinguished concepts, *Anything* and *Nothing*, in every ontology; the first one subsumes all the concepts in the ontology and the second one is subsumed by the rest of the concepts of the ontology (they are the top and the bottom of the ontology). Analogously, roles *Anyrole* and *Norole* are the top and bottom, respectively, of any role hierarchy.

3. Query Processing in OBSERVER

In the strategy proposed for query processing in Global Information Systems the following steps are remarkable (see Figure 2): *Query Construction*, *Access to Underlying*

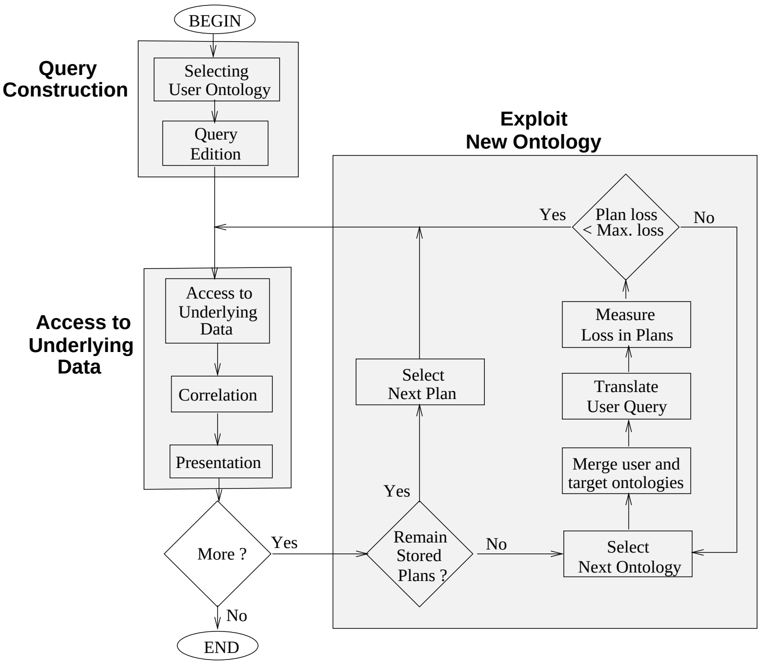

Figure 2: Detailed Query Processing in OBSERVER

Data and, if the user wants to enrich the answer, *Exploitation of a New Ontology*. Last two tasks are explained in detail in Sections and , respectively. All the process will be illustrated with an example in which the user wants to get an answer to the next query:

'Get the number of pages of documents which are periodical technical manuals written by only one author which is some kind of organization'

Query Construction

In this step two main tasks take place:

1. Selecting the User Ontology. The user browses the ontologies available in the Global Information System looking for an ontology that contains all the terms needed to express the exact semantics of her/his information needs. The chosen ontology will be called the *user ontology*. In the example, the Stanford-I ontology is selected (see Appendix), to express the query since it contains all the terms needed to express the semantics of the query.

2. Query Edition. By using a graphical tool, the user chooses terms from the user ontology to build the constraints and projections that compound the query:

   [number-of-pages] for (`AND` *document periodical-publication technical-manual (*`ATLEAST` *1 doc-author-name) (*`ATMOST` *1 doc-author-name) (*`ALL` *doc-author-name organization))*

   Moreover, a *defined term* (see Section) Q corresponding to the query is created in the user ontology.

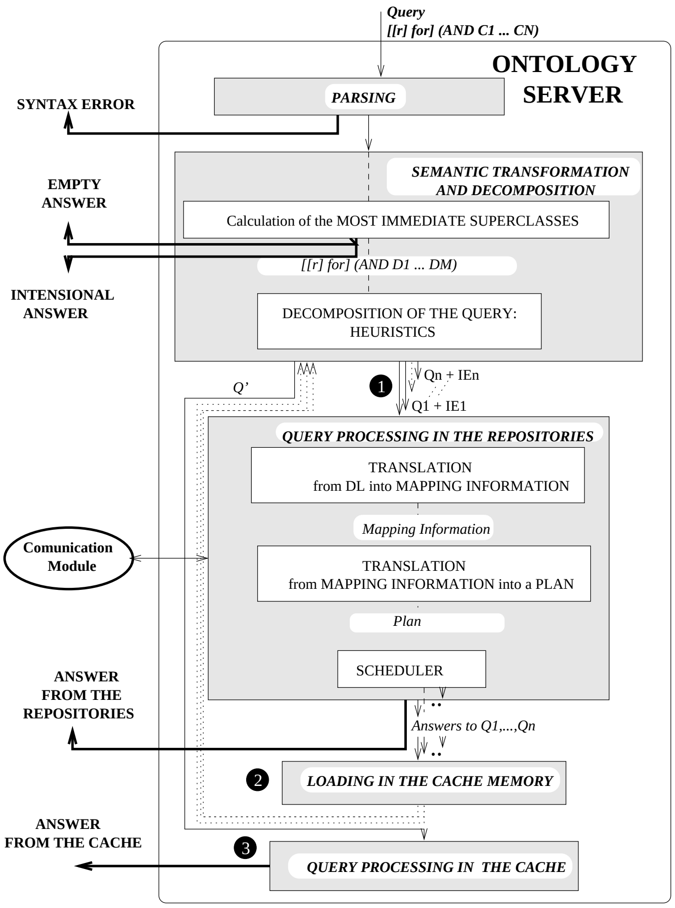
Query
[[r] for] (AND C1 ... CN)
ONTOLOGY SERVER
SYNTAX ERROR
PARSING
EMPTY ANSWER
SEMANTIC TRANSFORMATION AND DECOMPOSITION
Calculation of the MOST IMMEDIATE SUPERCLASSES
[[r] for] (AND D1 ... DM)
INTENSIONAL ANSWER
DECOMPOSITION OF THE QUERY: HEURISTICS
Q'
1
Qn + IEn
Q1 + IE1
QUERY PROCESSING IN THE REPOSITORIES
TRANSLATION from DL into MAPPING INFORMATION
Mapping Information
Comunication Module
TRANSLATION from MAPPING INFORMATION into a PLAN
Plan
SCHEDULER
ANSWER FROM THE REPOSITORIES
Answers to Q1,...,Qn
2
LOADING IN THE CACHE MEMORY
ANSWER FROM THE CACHE
3
QUERY PROCESSING IN THE CACHE

Figure 3: Steps for accessing data underlying an ontology.

4. Access to Data Underlying an Ontology

In this section we explain the way in which the Ontology Server accesses underlying data repositories. Five main stages are followed by the Ontology Server: parsing of the query, semantic transformation and decomposition, query processing in the underlying repositories, loading of the answers brought from the repositories in the cache memory and query processing in the cache memory. In Figure 3, those steps and the relationships among them are showed.

During the parsing, lexical and syntactical errors are detected. In our case this is achieved by using the parser of the DL system.

4.1. Semantic Transformation and Decomposition

The semantic transformation consists on finding a semantically equivalent query to the user query. For that, the set of *Most Immediate Superconcepts* ($\mathcal{MIS}$) is calculated (see [13]).

Theorem: The concept description of the query $\mathcal{C}$, is semantically equivalent to the intersection of all the immediate superconcepts in the set $\mathcal{MIS}$. The demonstration appears in [13].

In the semantic transformation and decomposition stage, different situations may happen: a) the query is detected as inconsistent (the empty answer is shown to the user) and therefore the process ends, b) the query is answered from the cache memory (number 3 in Figure 3) but first, if needed some subqueries are asked in the underlying repositories (numbers 1 and 2) or c) the query is answered directly from the underlying repositories and nothing is loaded in the cache memory (number 1 in the Figure 3).

When working with DL systems, it is also possible to give intensional answers, that is, answers in terms of the descriptions that the objects that form the extensional answer satisfy. Two different types of intensional answers are possible: *Most Specific Formulation* (MSF) of the query and *Extended Formulation* (EF) of the query. The first one is formed by the elements of the $\mathcal{MIS}$ set corresponding to the query and the second one is formed by recursively substituting the concepts in the initial query by their definitions. Both types of intensional answers (MSF and EF) are offered to the user by request but, in particular, the EF is also offered when the initial query is inconsistent, because it becomes more explicit where the inconsistency is.

Cache optimization

Dealing with ontologies connected to several data repositories, it is worth having some data cached within the ontologies in order to avoid accessing the underlying repositories each time a user formulates a query. There is a cache memory for each node where an ontology exists. This cache memory only stores objects and role values for terms existing in its corresponding ontology. Communications costs involved in transferring intermediate results and the final reconstruction of the answer can thus be avoided. So, if some data are cached, during the query processing it is necessary to detect if the query can be answered with the data stored in the cache, that is, if the query *is contained* in the cache (query completeness).

Luckily, when working with a DL system, the subsumption mechanism of concepts can be used to verify if queries are cached.

The test for query completeness is the next one:

A query is cached if all the concepts whose names appear in the set $\mathcal{MIS}$ corresponding to the query are explicitly or implicitly cached and all the roles that appear in the set $\mathcal{MIS}$ and the roles to be projected in the query are also cached.

The definitions of the notions of explicit and implicit caching that appear in the previous test are:

- *A concept C that belongs to the ontology, is explicitly cached by storing in the cache memory the objects that such a concept represents.*
- *A role is explicitly cached for a concept by storing their corresponding values in the cache memory. A role can be explicitly cached only for concepts that are explicitly cached.*
- *A query is implicitly cached, if all the concepts whose names appear in the set $\mathcal{MIS}$ corresponding to the query are explicitly or implicitly cached and all the roles that appear in the set $\mathcal{MIS}$ and the roles to be projected in the query are also cached.*

Query Decomposition

If a query cannot be completely answered with the contents of the cache memory then it must be decomposed. Query decomposition implies to obtain and analyze all the possible combinations of subqueries that can be made on the underlying repositories in order to get the query answer. However, in general the previous process is very complex because there can be many different ways of decomposing a query in subqueries[2] and statistic information and access paths in the local systems are not available. For that reason we have defined a set of heuristics that try to reduce such number of possibilities. These heuristics take into account if some parts of the semantically equivalent query to the initial one are cached, domain semantics and statistics about queries previously formulated. The goals of these heuristics are:

- Goal 1: To avoid that the answer sent from each repository is too large.
- Goal 2: To try that the computation cost in each repository is small, unless it is needed to reach the goal 1.
- Goal 3: To try not to bring parts that are already cached, unless it is needed to reach the previous goals.
- Goal 4: To try to send subqueries that are executed in parallel in different repositories, if these subqueries satisfy the two first goals, in order to avoid communication cost among the repositories and a greater computation cost in them.

The heuristics are presented in table 1.

Table 1: Heuristics used during Query Decomposition step.

H.	Goals	Description
H1	1,2,3	Substitute a non-cached defined concept without alternative[3] mapping information by its most specific definition
H2	1,2	Maintain a non-cached defined concept with alternative mapping
H3	1,3	Substitute a non-cached concept by one of its non-cached subconcepts only if the rest of the subconcepts are all cached
H4	1,3	Not to send a subquery with projection of a role that is already cached
H5	1	Reduce the size of the answer for a subquery within a repository
H6	4	Merge subqueries made over different repositories in order to reduce the size of the answer
H7	2	Transform a subquery with several restrictions over the same role by a subquery with the projection of that role

A more detailed explanation of these heuristics goes out of the scope of this paper.

[2] If the query is composed by n terms, there are as much possible decompositions as the number of possible partitions in a set of cardinality n. Moreover, the possibilities are greater because a term can appear in more than one subquery and a defined term can be substituted by its definition.

[3] Concepts and roles may have more than one mapping information to which we call alternative mapping informations.

4.2. Query Processing in the repositories

The set of subqueries that have been selected in the previous stage have to be asked in the underlying repositories. Before generating the database queries or application programs that retrieve the data, it is convenient to optimize the mapping information that has been expressed in a language independent of the access languages of the underlying repositories: the multidatabase extended relational algebra. The idea of the mapping information is *to view a data repository as a set of relations and attributes, independently of the concrete organization of the data in the repository.* Therefore, in this stage, for each DL subquery its corresponding optimal mapping information is generated. This optimization appears completely detailed in [11].

4.3. Loading and Query Processing in the Cache Memory

This stage is only performed when it is decided in the semantic transformation and decomposition stage. If it is needed, some subqueries are asked in the repositories under the ontology and then, the objects and role values that form the answer are loaded in the cache memory as objects (or role values) of new concepts added to the ontology (but that are not visible to users).

For example, the objects that form the answer for next subquery:

(`AND` *magazine (*`ATLEAST` *1 doc-author-name) (*`ATMOST` *1 doc-author-name))*

can be loaded in a new defined concept *magazine#1* whose description is exactly the previous subquery.

Once all the answers to the subqueries are loaded in the cache memory then, the semantically equivalent query to the initial one is asked in the cache memory and the answers for that query are the answers for the initial query. As it can be seen the correlation of the answers coming from the different data repositories is performed in the cache memory.

4.4. Example

In figure 4 we show briefly the different query processing steps for the query presented in section :

[number-of-pages] for (`AND` *document periodical-publication technical-manual (*`ATLEAST` *1 doc-author-name) (*`ATMOST` *1 doc-author-name) (*`ALL` *doc-author-name organization))*

5. Exploit a New Ontology

If the user is not satisfied with the answer, the Query Processor will retrieve more data from another ontology to "enrich" the answer. Notice that the answer is given in an incremental way. Therefore, the first task consists on selecting a component ontology, that we call *target ontology.* Once a target ontology is selected, the remaining task is to rewrite the user query in terms of that target ontology. Three main tasks take place: *Integration of user and target ontologies*, *Translation of the user query* and *Measure of Loss of Information.*

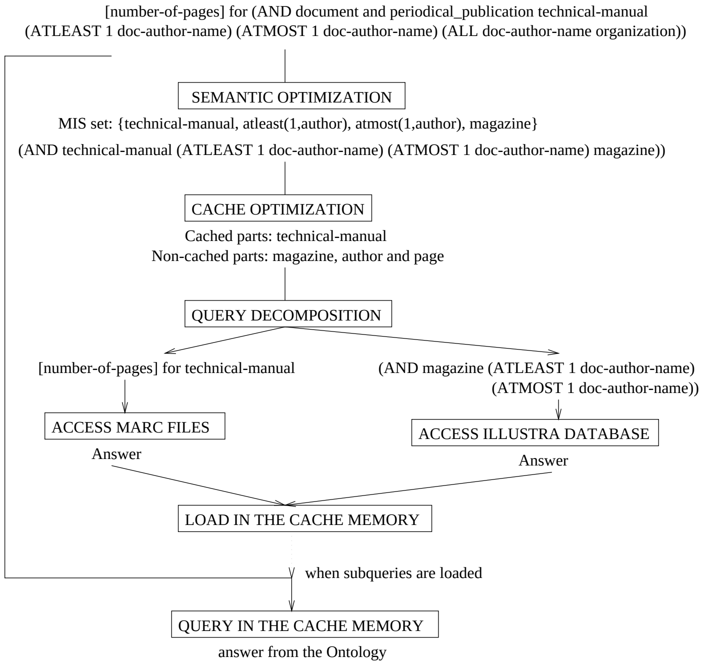
[number-of-pages] for (AND document and periodical_publication technical-manual
(ATLEAST 1 doc-author-name) (ATMOST 1 doc-author-name) (ALL doc-author-name organization))
SEMANTIC OPTIMIZATION
MIS set: {technical-manual, atleast(1,author), atmost(1,author), magazine}
(AND technical-manual (ATLEAST 1 doc-author-name) (ATMOST 1 doc-author-name) magazine))
CACHE OPTIMIZATION
Cached parts: technical-manual
Non-cached parts: magazine, author and page
QUERY DECOMPOSITION
[number-of-pages] for technical-manual
(AND magazine (ATLEAST 1 doc-author-name)
(ATMOST 1 doc-author-name))
ACCESS MARC FILES
Answer
ACCESS ILLUSTRA DATABASE
Answer
LOAD IN THE CACHE MEMORY
when subqueries are loaded
QUERY IN THE CACHE MEMORY
answer from the Ontology

Figure 4: Example of query processing performed by the Ontology Server.

5.1. Integration of user and target ontologies

Rewriting user query terms implies to deal with terminological relationships, stored in the IRM, between terms of different ontologies. However, DL systems provide functions that calculate terminological relationships between term descriptions in an ontology. In fact, we would need a DL system dealing with distributed ontologies and there does not exist any DL system with that feature.

The solution seems to be the integration of the user and the target ontology taking advantage of the deductive power of the DL system [4]. Properties between terms in both ontologies are exactly the interontology relationships stored in the IRM[4], so no intervention of the user is needed. Although some of the previous relationships can be redundant the DL system will classify the terms in the right place of the integrated ontology. To know if the resulting terms are *primitive* or *defined* we apply the rules described in [5].

Following with the example, and taking as target ontology WN (see Appendix), the result of the integration of the user and the target ontology by applying the relationships defined between them is shown in Appendix . Any term without any parent in the figure is really a subconcept of *Anything.* Terms from the ontology WN are in uppercase and terms from Stanford-I are shown in lowercase; you can see that the user query, Q, has been also classified by the DL system in the right place.

5.2. Translation of the user query

Once the integrated ontology is built, the only task that the system has to perform is to fully translate Q (already simplified semantically) into terms labeled as terms from the target ontology (in the example, those in lowercase). Hereafter, the Query Processor will only deal with the integrated ontology since it contains all the needed information.

[4] There can exist semantic relationships among ontologies describing different domains.

Any conflicting term in the user query (those with no synonym in the target ontology) is substituted by the intersection of its immediate parents/hypernyms or by the union of its immediate children/hyponyms. This method is applied recursively until a full translation of the conflicting term is obtained. Moreover, every expression obtained is simplified (using DL systems capabilities) in order to eliminate redundant subexpressions. Notice that it is always possible to get at least one full translation of any conflicting term in both directions since terms *Anything* and *Nothing* (see Section) are terms from the target ontology (they always exist in any ontology).

Traversing hyponym and hypernym relationships can result in several possible translations. All the possibilities are explored and the result is a list of tuples in the format $< Plan, Loss >$, where *Plan* is an expression in Description Logics using only terms from the target ontology, and *Loss* is a number between 0 and 100 representing the percentage of loss of information of *Plan* with respect to Q. The loss of information of a plan is calculated after obtaining all the possible plans (see Section). At the end, that plan with less loss of information is chosen to enrich the answer to the user.

After all the possible plans are generated, they are optimized by removing the redundant plans following this rule:

$$< Plan1, Loss1 > \subset < Plan2, Loss2 > \Leftrightarrow Plan1 \subset Plan2 \wedge Loss1 > Loss2$$

In such a case, $< Plan1, Loss1 >$ will be eliminated. If the first condition if not satisfied, Plan1 could bring new relevant objects; if the second one is not satisfied, Plan1 will be chosen before Plan2 before it has less loss. After optimizing plans, the one with less loss is chosen to access new relevant data. The rest of the plans are stored and could be used if the user still wants more data.

Let us see now the resulting plans that are obtained in our example. Remember that the target ontology is WN and the query built using terms of Stanford-I, after semantic transformation, is the following:

$$Q = \textit{[number-of-pages] for (}\texttt{AND}\textit{ magazine periodical-publication technical-manual}$$
$$\textit{(}\texttt{ATLEAST}\textit{ 1 doc-author-name) (}\texttt{ATMOST}\textit{ 1 doc-author-name)}$$

After integrating Stanford-I and WN ontologies Q is rewritten in the following way:

$$Q = \textit{[PAGES] for (}\texttt{AND}\textit{ MAGAZINE technical-manual (}\texttt{ATLEAST}\textit{ 1 CREATOR)}$$
$$\textit{(}\texttt{ATMOST}\textit{ 1 CREATOR)}$$

We can observe that the only term without translation is 'technical-manual' due to the non-existence of a synonym in WN of such a term. Then, the only way to obtain a full translation of Q is to substitute the conflicting term by a combination of term parents either a combination of term children. As 'technical-manual' has no children the only way is upwards, i.e., to substitute 'technical-manual' by the intersection of its parents. In this case, the only parent is 'MANUAL' which is a term of WN, the target ontology. Therefore, the only plan to translate Q completely into terms of WN ontology is the following:

$$< \textit{[PAGES] for (}\texttt{tt}\textit{ MAGAZINE MANUAL (}\texttt{ATLEAST}\textit{ 1 CREATOR)}$$
$$\textit{(}\texttt{ATMOST}\textit{ 1 CREATOR)}, ? >$$

5.3. Measure of Loss of Information

The change in semantics caused by the use of hyponym and hypernym relationships must be measured not only in order to decide which substitution minimizes the loss of information but also to present to the user some kind of "level of confidence" in the new answer. The loss of information can be measured in an extensional manner and in an intensional manner.

We use measures based on the underlying extensions of the terms in the ontologies in order to measure the extensional loss of information. That information can be obtained and updated by batch processes so that it is available when needed for answering queries. We propose an adaptation of well established measures like *precision* and *recall* to measure the information loss when a term is translated by its hyponyms or hypernyms. A composite measure combining precision and recall [23] is used to choose a translation with the least information loss. See [19] for more details about this measures.

Moreover, using the DL systems capabilities an intensional loss based on the terminological difference can also be calculated.

Following with the example, the loss of information of the plan obtained in the previous subsection is calculated based on the extensions of the terms 'technical-manual' from Stanford-I and 'MANUAL' from WN ontology. We use a parameter called 'user-precision' which is a percentage describing the relevance for the user of precision over recall; we assume in the example that it is 50%.

$Ext(technical - manual) = 19, Ext(MANUAL)^{5}$=1

$$\text{Precision.low}=\frac{Ext(technical-manual)}{Ext(technical-manual)+Ext(MANUAL)}=0.95$$

$$\text{Precision.high}=\frac{Ext(technical-manual)}{max[Ext(technical-manual),Ext(MANUAL)]}=1$$

$$\text{Recall}=1$$

$$\text{Loss.low}=1\text{-}\frac{1}{\frac{user-precision}{Precision.high}+\frac{user-recall}{Recall.high}}=0$$

$$\text{Loss.high}=1\text{-}\frac{1}{\frac{user-precision}{Precision.low}+\frac{user-recall}{Recall.low}}=0.025$$

Finally, the data underlying the previously presented plan is accessed. The information retrieved must be correlated with the previously presented to the user, in order to remove redundant information. After correlation, the new answer is presented to the user with the following explanation:

> *"The answer has been updated with information underlying WN ontology. The new answer is composed of magazines which are manuals with exactly one author. They are technical manuals (as the user requested) with a probability between 97.5% and 100%"*

6. Conclusions and Future Work

We have presented in this paper a new strategy based on information content for querying heterogeneous and distributed data repositories. All the followed steps for the query processing have been explained briefly. Moreover we want to mention the existance of a prototype that allows accessing different heterogeneous data sources in the domain

[5] Although 'MANUAL' subsumes semantically 'technical-manual' as they belong to different ontologies/data repositories their extensions can not satisfy the property but our measures take this into account.

of bibliographic references, which is only an example application domain. Both data repositories and ontologies describing them have been designed by other working groups and organizations. A complete description of ontologies and data repositories can be found in [20]. It is important to notice the heterogeneity among the ontologies (semantic heterogeneity) as well as in the data repositories (structural heterogeneity, different data structures: plain files, databases, WWW documents, etc,) and operational heterogeneity (some data repositories are accessed using SQL commands, others by WWW browsers, and some of them they do not even have a defined query language or access method) because they have been developed by different organizations. In this way we want to capture a real case and deal with problems that never arise when ontologies and data repositories are designed under the same point of view.

The prototype has been developed using last technology in programming techniques: fully implemented in Java (*applets* and *servlets*), CORBA is the communication protocol and VRML has been used to provide a 3D representation of ontologies. It is accessible for WWW browsers at http://siul02.si.ehu.es/ jirgbdat/OBSERVER. We are developing several graphical interfaces to help users and administrators of ontologies, data sources and IRM.

References

[1] E. Achilles, B. Hollunder, A. Laux, and J. Mohren. KRIS: Knowledge Representation and Inference System. Technical Report D-91-14, DFKI Kaiserslautern-Saarbrucken, 1991.

[2] Altavista. http://www.altavista.digital.com.

[3] Y. Arens, C.A. Knoblock, and W. Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6(2-3):99–130, 1996.

[4] J. M. Blanco, A. Illarramendi, A. Goñi, and J. M. Pérez. Using a terminological system to integrate relational databases. In *Information Systems Design and Hypermedia*. Cepadues-Editions, 1994.

[5] J.M. Blanco, A. Illarramendi, and A. Goñi. Building a Federated Database System: An approach using a Knowledge Based System. *International Journal on Intelligent and Cooperative Information Systems*, 3(4):415–455, December 1994.

[6] A. Borgida, R. Brachman, D. McGuinness, and L. Resnick. CLASSIC: A structural data model for objects. In *Proceedings of ACM SIGMOD-89*, 1989.

[7] R. Brachman and J. Scmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2), February 1985.

[8] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of the 10th IPSJ, Tokyo, Japan*, 1994.

[9] C. Collet, M. N. Huhns, and W. Shen. Resource integration using a large knowledge base in CARNOT. *IEEE Computer*, pages 55–62, December 1991.

[10] D. Florescu, L. Raschid, and P. Valduriez. A methodology for query reformulation in CIS using semantic knowledge. *International Journal of Cooperative Information Systems*, 5(4):431–467, 1996.

[11] A. Goñi, J.M. Blanco, and A. Illarramendi. Connecting knowledge bases with databases: a complete mapping relation. In *Proc. of the 8th ERCIM Workshop. Trondheim, Norway*, 1995.

[12] A. Goñi, A. Illarramendi, and E. Mena. Semantic query optimization and data caching for a multidatabase system. In *Proceedings of the Basque International Workshop on Information Technology.* IEEE Computer Society Press, July 1995.

[13] A. Goñi, A. Illarramendi, E. Mena, and J.M. Blanco. An optimal cache for a federated database system. *Journal of Intelligent Information Systems*, 9(2), 1997.

[14] T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition, An International Journal of Knowledge Acquisition for Knowledge-Based Systems*, 5(2), June 1993.

[15] Infoseek. http://www.infoseek.com.

[16] B. Kahle and A. Medlar. An Information System for Corporate Users : Wide Area Information Servers. *Connexions - The Interoperability Report*, 5(11), November 1991.

[17] A.Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, 5(2):121–143, September 1995.

[18] R. MacGregor. A deductive pattern matcher. In *Proceedings AAAI-87*, 1987.

[19] E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth. Scalable query processing in dynamic and open environments: An approach based on information brokering across domain ontologies. Submitted for publication, 1997.

[20] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. In *Proc. of the First IFCIS International Conference on Cooperative Information Systems (CoopIS'96), Brussels (Belgium), June.* IEEE Computer Society Press, 1996.

[21] S. Milliner and M. Papazoglou. Scalable information elicitation in large heterogeneous database networks. To be published in IEEE Internet Journal, 1997.

[22] R. Bayardo, W. Bohrer, R. Brice, A. Cichocki, G. Fowler, A. Helai, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. Infosleuth: Semantic integration of information in open and dynamic environments. In *Proceedings of the 1997 ACM International Conference on the Management of Data (SIGMOD), Tucson, Arizona.*, May 1997.

[23] C. J. van Rijsbergern. *Information Retrieval*, chapter 7.
http://dcs.glasgow.ac.uk/Keith/Chapter.7/Ch.7.html.

[24] K. von Luck, B. Nebel, C. Peltason, and A. Schmiedel. The anatomy of the BACK system. Technical Report KIT Report 41, Technical University of Berlin, Berlin, F.R.G., 1987.

Appendix A. Ontologies

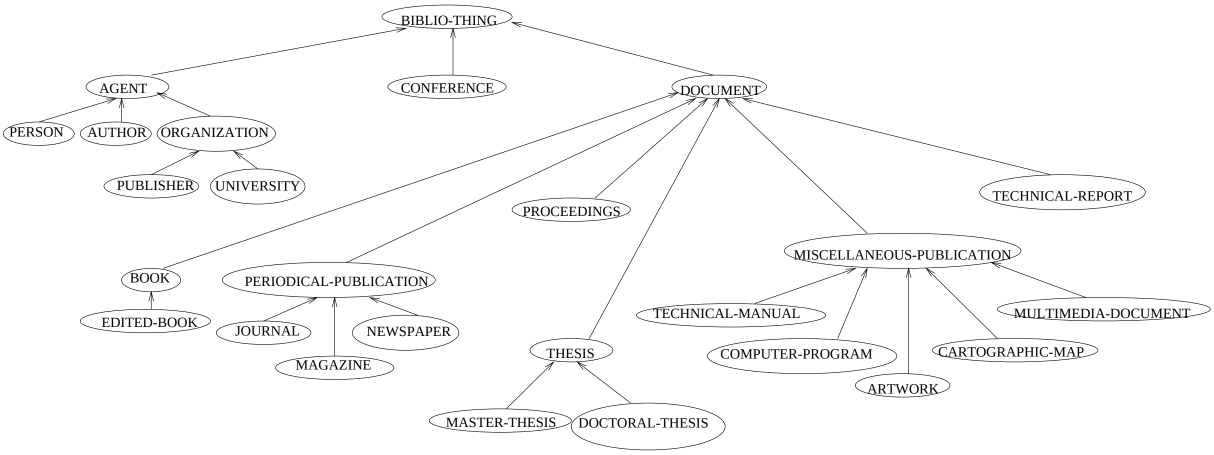

Figure 5: Stanford-I: A subset of the Bibliographic-data ontology

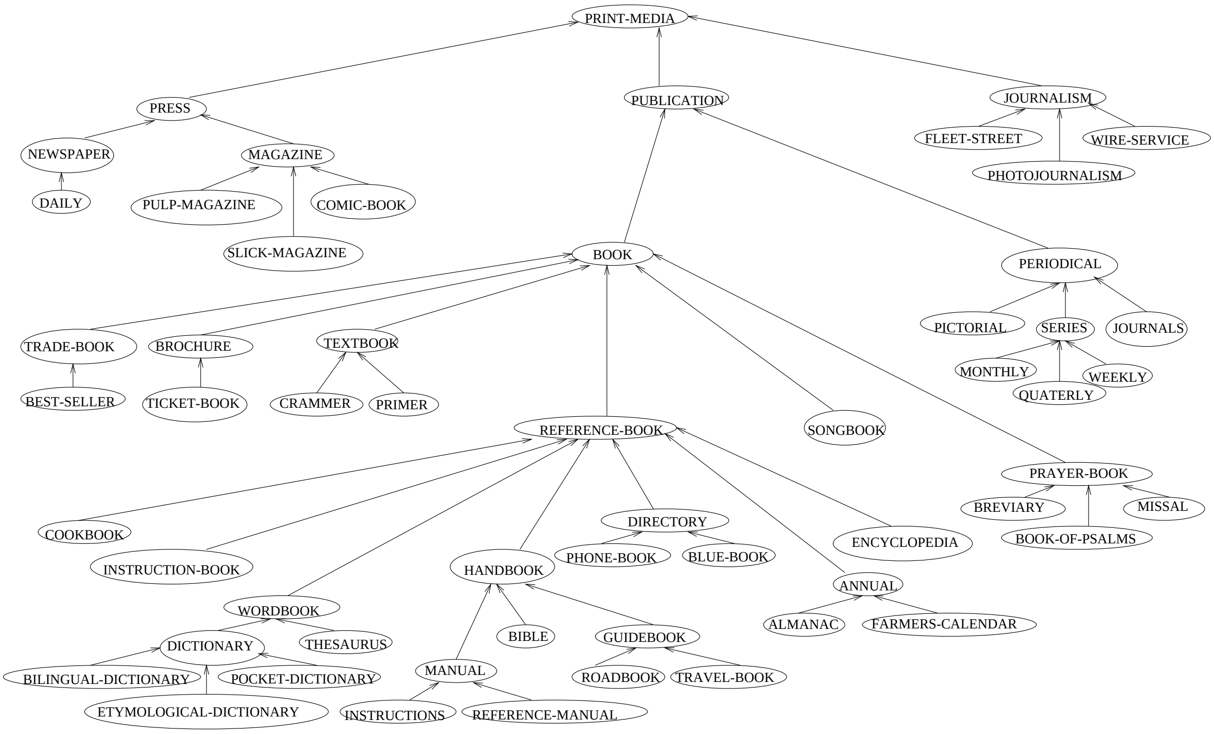

Figure 6: WN: A subset of the WordNet ontology

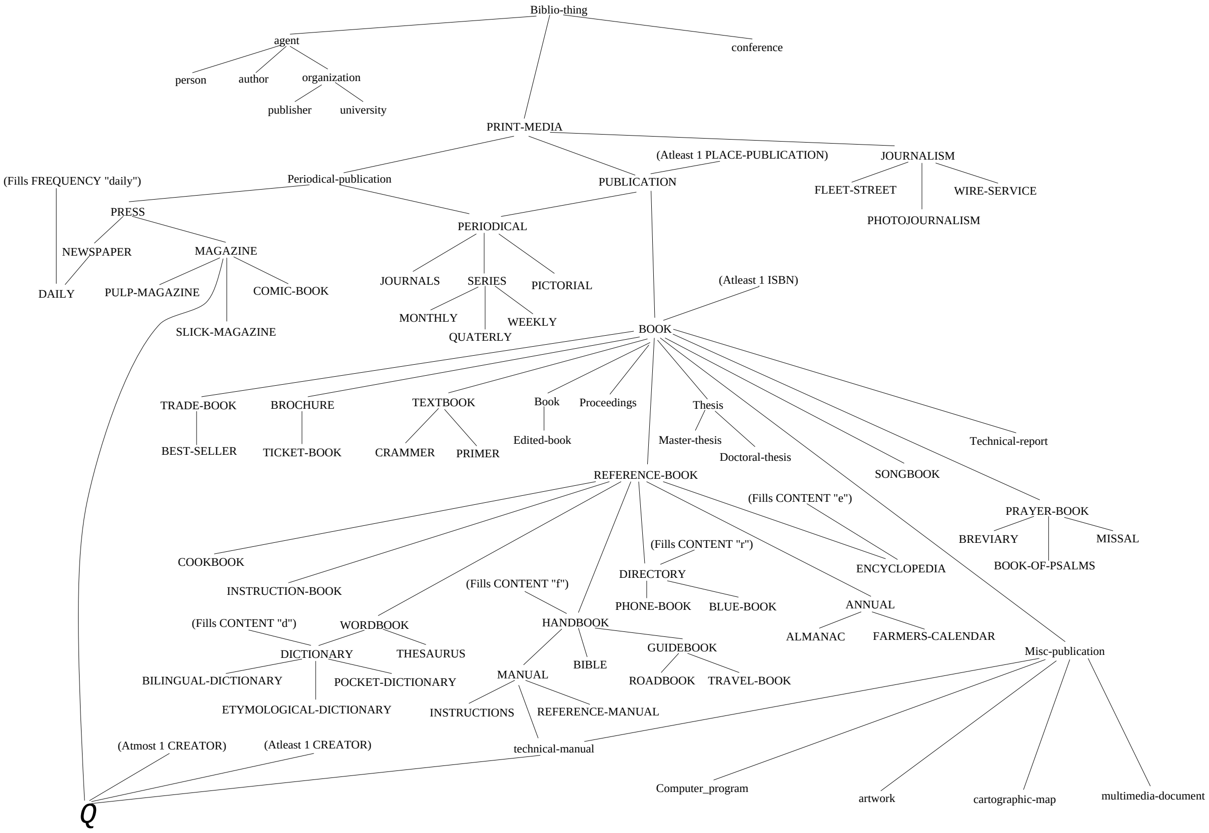

Figure 7: Integration of Stanford-I and WN ontologies