

# Query Processing on the Semantic Web

*Heiner Stuckenschmidt, Vrije Universiteit Amsterdam*

## **Introduction**

Methods for query processing are an essential part of database and information systems (see e.g. [Garcia-Molina et.al. 2002]). Named queries, also called views, are not only used to access information, but also for (re-)structuring and integrating information [Halevy 2001]. On the World Wide Web, effective query processing used to be impossible due to the lack of data structures and scheme information. The ability to efficiently access information in a query-like fashion had been sacrificed for the ease of authoring and publishing information.

When we talk about the Semantic Web today, then we mainly refer to an effort of bringing back structure to the Information that is available on the World Wide Web. This time, structures do not come in the shape of well-defined database schemas but in terms of semantic annotations that conform to a specific, often loosely defined schema or even to an explicit specification of the intended meaning of a piece of information, called ontology. The first real results of semantic web research are languages for encoding these three components: The resource description framework RDF provides a language to encode semantically annotated information, the RDF vocabulary (formerly RDF schema) [Brickley and Guha 2002] is a language for capturing schema information in terms of a typing system with inheritance and typed relations. Finally, the Ontology Web Language OWL [McGuinness and van Harmelen 2002] is a logical language that can be used to describe ontologies that further constrain the possible interpretations of terms used to annotate information.

This return of structure to the World Wide Web enables us to re-consider the issue of query processing. Having a closer look at the nature of the information on the web and the languages used to impose structure on this information, we have to recognize that conventional database techniques are not sufficient to make query effective and efficient processing on the Semantic Web possible. This paper is devoted to providing evidence for this claim. We successively discuss the specific characteristics of the semantic web, that force us to consider new techniques for query processing. We base the discussion of these characteristics on experience gained when experimenting and theoretically investigating the issue of query processing on the semantic web. We mention some approaches to overcome existing problems whenever they exist.

In section 2 we discuss the problem of querying ontology-based information and the need for logical reasoning in order to achieve complete answers. Section 3 introduces the problem of heterogeneity arising from the use of different ontologies and the need to integrate ontologies for answering queries. The problem of incompleteness of information is discussed in section 4 and the use of approximation techniques in querying is motivated. Discussing scalability, section 5 raises the most obvious and still most striking problem of the semantic web. We argue for need to transform representations of information in order to be able to apply more efficient querying methods. Section 6 addresses the problem of the dynamic character of information on the web where information sources constantly occur and disappear forcing us to perform a discovery step in order to locate useful information sources before actually being able to query information. We conclude with a summary of the problems arising when trying to query the semantic web and an attempt to formulate the most pressing research questions in this direction.

## Querying needs Deduction

In Databases, query processing is mostly based on formalisms that assume a closed world and use the principle of negation as failure. On the semantic web, such an assumption is clearly not adequate as we always have to deal with incomplete knowledge. The way the semantic web deal with this incompleteness is the use of background knowledge in terms of ontologies the information is related to. These ontologies specify constraints that must hold in all possible states of the world and therefore reduce the uncertainty about its real state. Figure 1 shows a simple ontology that might be used to provide background knowledge about software components on the web.

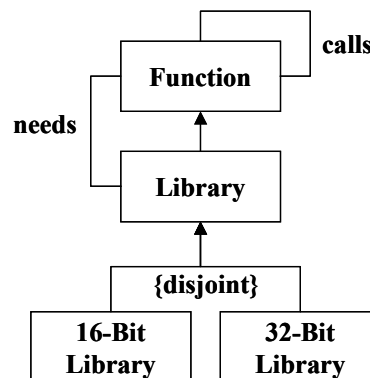


Figure 1: A Simple Ontology

The ontology specifies classes of objects, inheritance relations between these classes and typed relations between classes. Further, it specifies that the class *Library* is a partition of 16 Bit and 32 Bit libraries. The classe and relation names in this hierarchy can now be used to provide information about objects of the web by assigning them to a particular class and defining relations between them. The structure of the ontology now adds constraints that have to be considered when asking queries about these objects. As we assume that the information is described using only terms from the ontology, queries we ask about the information also have to be phrased using this vocabulary. We consider the case where we want to find out whether a piece of software implicitly uses a 16-Bit library (if this is the case we might not want to use this specific software). We use the following query to get the required information:

$$\begin{aligned}
 Q(X) \leftarrow \\
 & \text{calls}(X, Y) \wedge \text{32-Bit-Library}(Y) \wedge \\
 & \text{needs}(Y, Z) \wedge \text{16-Bit-Library}(Z)
 \end{aligned}$$

Computing the answer to this query seems to be straightforward. We just have to find a function and check whether it is connected to a 32-Bit-Library by the calls relation which in turn is connected to a 16-Bit Library. If we indeed find such an object, we can be sure that it is an answer to the query. In the case where an object does not explicitly have these properties, we are not allowed to conclude that it is not an answer to the query as this would correspond to the closed world assumption, we have to drop on the semantic web. We rather have to check all possible states of the world using deductive reasoning. We assume a true state of the world as depicted in Figure 2. This example illustrates the difference between closed world reasoning and open world reasoning with background knowledge. We want to find out whether the Function *gui.exe* is an answer to our query. Using conventional query answering techniques, the answer would be 'no', because we are not able to decide whether *dosSim.dll* is a 16-Bit or a 32-Bit Library causing the query to fail due to lack of information.

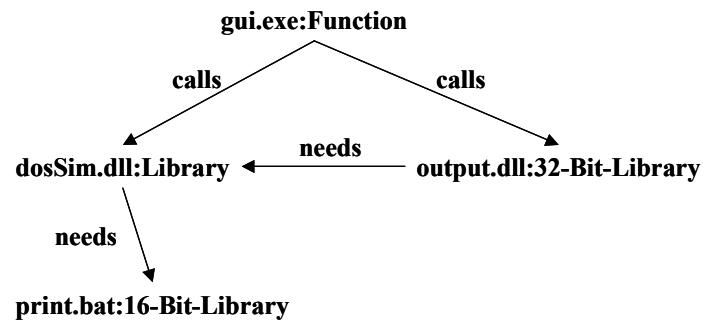


Figure 2: An Incomplete Information Source

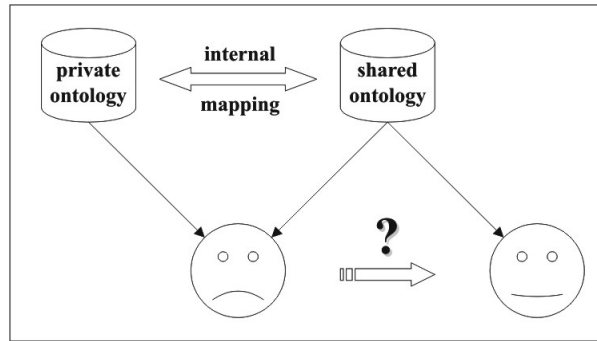
In the presence of background knowledge in terms of the ontology of Figure 1, however, the answer would be ‘yes’, because the conditions are satisfied in all possible states of the world. This claim is supported by the fact that `dosSim.dll` either has to be a 16-Bit or a 32-Bit Library, not both and not none of them. In the first case, the left hand path of the graph supports the requirement of the query, in the second case, the triangular path at the top of the graph does.

A solution to this problem of open world querying with background knowledge is the reduction to a satisfiability problem in description logics that can be solved by Tableau-style proof procedures. Horrocks and Tessaris propose such a translation by ‘rolling-up’ conjunctive queries into concept expressions [Horrocks and Tessaris 2002] and prove the correctness and completeness of their approach with respect to the task of deciding whether an object is an answer to a given question [Horrocks and Tessaris 2000].

### ***Querying needs Integration***

The deductive query answering approach by Horrocks and Tessaris can be used to answer queries about information that conforms to one single ontology. On the semantic web, however, there will not be a single ontology all information sources use. Today, there are already more than a hundred ontologies available for public access only in the DAML ontology library and an increasing number of users participating in the semantic web, this number will increase significantly. In the long term we will face a situation, where large numbers of software agents populate the semantic web each using a number of ontologies as a basis for its communication which includes queries posed to other agents. If the agent that receives the query wants to compute the answer, it first has to find an alignment between the vocabulary used in the query expression and its own vocabulary that links the available information to background knowledge. There are different options for relating ontologies.

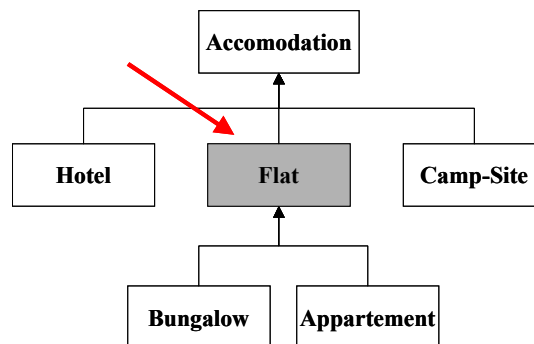
In real life situations, the integration problem is often weakened by the fact that there are already ontologies about many topic areas available. As building ontologies is a very time consuming and tedious task, designers of ontology-based systems will often use these existing ontologies as a part of the background knowledge and supplement it by additional definitions specific for that particular application rather than building the complete background theory from scratch. This leads to a situation, where different agents on the semantic web use partially overlapping background knowledge. Further, the non-shared part of the ontologies of different systems will be integrated with the shared part, because the systems need to use the background knowledge as a whole. If one agent wants to query another one, what he has to do is to determine the overlapping part of background knowledge used by himself and the other agent and to re-phrase his query in using terms from the shared part.



**Figure 3: Integration with partially shared Ontologies**

We have shown that this translation can be done in an approximate way using technology, i.e. description logics and terminological reasoning [Stuckenschmidt 2002a]. The idea is to build an approximate classifier that determines the upper and lower bounds of a concept name in the shared part of the ontology. Instead of discussing the technical details that can be found in the referred paper, we give an example illustrating the approximation idea.

As an example of concept approximation we use the concept Flat. The relevant part of the overall hierarchy can be seen in Figure 4. The approximations we are interested in are the direct sub- and super-classes of our example concept that are not from the same ontology. We can see in the figure that these are: 'Bungalow' and 'Appartement'. Applying our approximation method, we decide that all members of these classes are assumed to be members of the concept 'Ferien-Wohnung'. While this result is not completely true, because houses are not flats, it still serves the purpose very well, because all of concepts describe accommodations that are reasonable replacements in the case that no flat is available.



**Figure 4: Coordinated Concept Hierarchy**

If we determine the upper approximation of the example concept, we get the general concept 'Unterkunft' (accommodation). Our method now determines all instances of this general concept to be potential members of the example concept. Besides the members of the already mentioned concepts, this also includes objects that are members of the concepts 'Hotel' and 'Camp-Site'. We see, that these results are still closely related to the example concept, because they are all accommodations mainly used during holiday, however, hotels and Camp-Sites are not really the kind of answer the user would assume to get when asking for a flat. Still, returning hotels and camp-sites as answers to a query for a flat is still better than not returning any result, because the user might want to change her choice in favor of other preferences (e.g. the location).

In the same way, we can compute approximate answers to conjunctive queries by transforming the query into a concept expression and approximating this concept expression in the way sketched above [Stuckenschmidt et.al. 2002].

Open issue with respect to the interpretation problem are the development and evaluation of reliable methods for finding the right mappings between ontologies and the adaption of current semantics of ontology languages to the distributed setting.

### **Querying needs Approximation**

In the course of a case study it turned out that in most cases the approximation of concept expressions returns good results, because people tend to share a reasonable number of concept names across different ontologies that provide a basis for creating mappings. These mappings can often be found using stemming and simple string matching. On the other hand, it turned out that it is much harder to come up with reasonable mapping between the relations used in different ontologies leading to a situation where we only have very sparse mappings between these relations. This in turn has a major impact on the quality of approximation applied to conjunctive queries. In the presence of sparse mappings, we face a situation where the descriptions in different ontologies referring to the same real-world object can be significantly different. In most cases, the descriptions are different in the sense that different relations are used to related same object to other objects in the domain. These relations may refer to the same properties of the object that cannot be matched due to a missing mapping or the set of properties itself used might be different. As a consequence, real-world objects that are meant to be an answer to a query are not returned because their description does not match the query that is formulated using terms from a different ontology. We address this problem by relaxing the query, i.e. by weakening those constraints from the query expression that are responsible for the failure. In order to be useful, this weakening process has to fulfill certain formal properties. In especially, we want to make sure that we do not loose any answers when modifying the query. We illustrate the relaxation process using the following query from a case study:

$$Q(X) \leftarrow \\ \text{Hotel}(X) \wedge \text{zimmer}(X, 25) \wedge \text{ist-in-schloss}(X, ja) \wedge \\ \text{liegt-in}(X, V) \wedge \text{Ort}(V) \wedge \text{liegt-in}(V, \text{mecklenburg})$$

There are many different ways of making a query more general in order to increase the chance of matching a potential answer. A possible approach is based on the fact that each variable in a conjunctive might fail to match a specific object if the object does not satisfy the constraints. Therefore, a way of increasing the chance of matching the target object in the head of the query is to successively eliminate non-answer variables from the query. In the example query for example, we have the variables V, W, X, Y and Z where X is the answer variable. Therefore we can weaken query by eliminating the variables V, W, Y and Z. This can be done by removing all conjuncts containing a specific variable from the query expression. It is easy to see that successively removing conjuncts from the query leads to a sequence of queries with the desired properties [Stuckenschmidt and van Harmelen 2002].

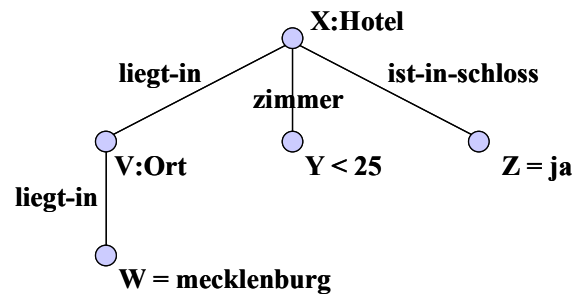


Figure 5: Variable Dependencies in the Query Graph

The main question that arises when adopting the variable elimination approach is the order in which the variables should be removed from the query. This order is partially constrained by the dependencies between the different variables. Removing the wrong variable first can break these dependencies and make remaining conjuncts useless. Looking at the example query this would happen if we first removed the variable  $Y$ . In this case the conjunct  $V = \text{mecklenburg}$  would be isolated, because the variable  $V$  only occurred in the removed conjuncts that connected it to the answer variable. In order to avoid breaking dependencies when removing conjuncts, we can use the query graph of the query to be relaxed as it explicates existing dependencies. In the query-graph dependencies between variables are represented by arcs between nodes. Therefore we have to ensure that the query graphs remains connected when removing the node that represents the variable we want to eliminate. Obviously, this is only the case if we eliminate variables that correspond to leaf nodes in the graph [Stuckenschmidt et.al. 2002].

This general framework for query approximation still contains many arbitrary choices. The crucial task for future research is how to design heuristics that guide the relaxation process. These heuristics might be based domain knowledge, user preferences and the task to be solved.

### **Querying needs Transformation**

Up to now, we only considered the problem, that different agents will use different vocabularies completely ignoring the way the semantics of these vocabularies are represented. A number of languages with sometimes very different features have been proposed for this purpose [Gomez-Perez and Corcho 2001]. When system designers choose a language for encoding information semantics, their choice is driven by various decisions. Different types of knowledge can be used for different kinds of reasoning tasks. Further different kinds of reasoning methods result in different levels of reasoning complexity. In order to choose a language, the designer has to take design constraints into account that originate from the application at hand. In particular, the choice of the language will be biased by the need of applying automated reasoning techniques. It is well known that there is a trade-off between the representational capabilities of the language used for describing knowledge and the ability to perform efficient reasoning. This of course also affects the problem of query answering, as all techniques discussed so far heavily rely on logical reasoning in some description logic. A first attempt to deal with the problems that arise from the representation-reasoning trade-off is the OIL ontology language [Fensel et.al. 2001] that consists of a family of languages of different complexity. While the purpose of the smallest language of the OIL family Core OIL is to define a well founded semantics for schemas of the emerging web standard RDF. This language is rather small and therefore allows efficient reasoning. The main language Standard OIL is tailored to have efficient reasoning support for consistency checking and for automatic construction of inheritance hierarchies for an extremely expressive logic. However this language does not include assertional language, because this

would disable the reasoning support. For applications where instances are required, the OIL defines the language Instance-OIL that includes instances, but has no reasoning support.

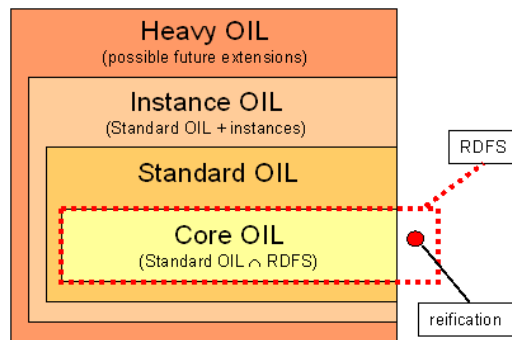


Figure 6: The Layered Architecture of OIL

The OIL framework allows a user to select between languages of different expressive power, however it does not address the problem of tailoring a language to a given application. Our main objective is that the current architecture of the OIL framework does only allow for strict extensions excluding the possibility to define alternative language that only partially overlap. As a response to this missing flexibility the ‘family of languages’ approach has been proposed [Euzenat and Stuckenschmidt, to appear]. The idea behind this approach is to have a set of languages structured by a partial order. This is more general than the layering approach and more convenient for the users who can find languages closer to their needs (or, for an intermediate language, languages closer to their own languages). In order to fulfill the translation task, for every two languages in the family a third language should exist that covers both of them. This third language is the one both languages can be translated into thereby achieving interoperability.

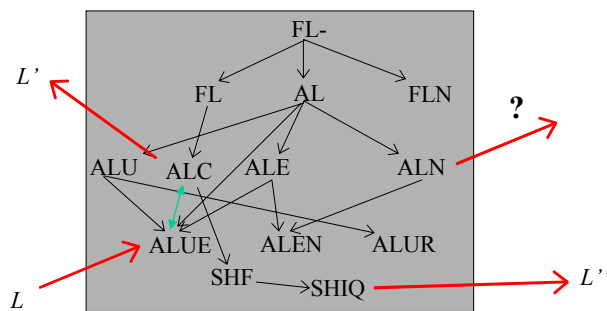


Figure 7: A Flexible Transformation Architecture

In fact, the family of languages approach is a generalization of the pivot and the layered approach that further increases the flexibility of the transformation process. The approach generalizes the pivot approach insofar as the pivot approach fulfills the family of languages property, because the pivot language can always be used as integration language. It also generalizes the layered approach, because in the layered framework the language that is higher in the hierarchy can be used as the integration language in the sense of the family of languages property. However, the family of languages approach is more flexible, because it does not require a fixed pivot language nor a fixed layering of language. On the contrary, any

language that fulfills certain formal criteria can be used as integration language (for details see [Euzenat and Stuckenschmidt, to appear]).

Obviously the problem of different capabilities does not only relate to the logical language used, but also for the reasoning services available. While one system might directly support ontology-based query processing, other systems might only support satisfiability testing. Using the same ideas discussed above, we can stepwise transform the query answering problem into a satisfiability problem.

### Querying needs Discovery

The main idea of the Semantic Web is that you cannot only use information from your own place, but the information of the whole web. This however requires that potential information sources are located. On a technical level this is done by the web infrastructure that provides information about addresses of different servers. A problem that remains is to select those sources that actually have relevant information we might want to query. For this purpose we have to develop a discovery method is in charge of finding information sources on the web that can solve (parts of) a particular query. The corresponding sources have to be listed and ranked according to the expected quality (in especially completeness) of the answer. The problem of locating information sources has been addressed in the area of information retrieval and filtering for a long time, however, the presence of background knowledge in terms of ontologies poses new challenges to the discovery task. A useful discovery method now also has to check

- Whether the vocabulary of that source is alignable with the query vocabulary
- Whether the representation of the background knowledge can be dealt with
- Whether there is a reasoning service available for actually computing the answer

In short, the discovery step has to take into account solutions of all the problems mentioned above.

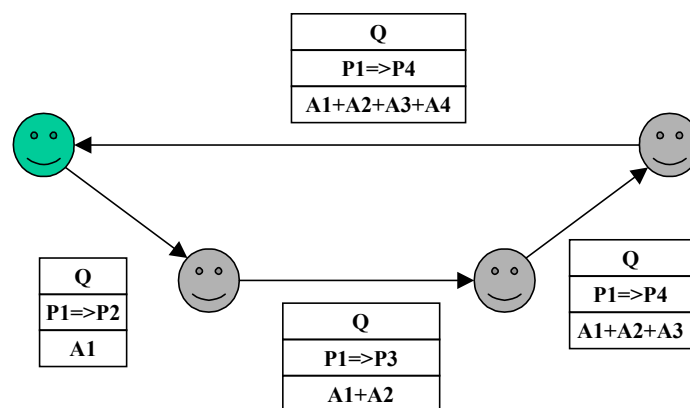


Figure 8: Integrated Query Answering and Expertise Discovery

In order to lower the entrance barrier to the semantic web on the cooperation level, agents might implement functionality for promoting their own expertise. This could be done by not only publishing the ontology on the web but also directly sending messages to other agents that contains a subtree of the agent's ontology, possibly along with some instance information. This information could serve as a basis for starting a meaning negotiation process where two agents try to find correspondencies between their ontologies that could be used as a basis for query answering later on. Besides having a way to actively promote information on the

network, agent may also implement methods for actively exploring knowledge on the web by explicitly asking other agents about their capabilities. This does not differ much from requesting specific information as most ontology-aware query languages support schema level queries that can be used to explore another agent's ontology. These capabilities may have to be extended by specific queries concerning the expertise of an agent that have to be developed in the project.

A more subtle way of facilitating the discovery of expertise on the web is illustrated in Figure 8. The idea is to integrate expertise discovery in the query answering process. The process of integrated query answering and expertise discovery is the following: The querying agent (P1) send his query (Q) to an agent he assumes to have further information about the specific topic, because related classes in the working model of P1 name this agent (P2) as their source. Along with the query P1 also send the mapping information (P1=>P2) he has in his internal model and the answers he gets from his information. The reason for also sending the mapping information is the fact that the complete mapping information is distributed between the working models of the two agents. Sending his own mapping information ensures that P2 now has the complete set of known mapping rules. A nice side effect is that P2 can use the mapping to extend his own model in the spirit of evolving semantics and is therefore rewarded for processing the query. The information about known results is sent in order to make sure that no duplicate answers are returned, because P2 can check which of his answers to the questions are indeed new ones. In the case that P2 knows about another agent that also has expertise about the area (say P3) he adds his answers to the answer set. Further, he combines his updated mapping rules (P1=>P2) with the mappings (P2=>P3) and passes the query on to P3 along with the grown answer set (A1+A2) and the extended mapping (P1=>P3). This new message is handled by P3 in the same way and is passed on until no further agents are found that may contribute to the query. The query is then returned to P1 along with a complete answer set and mapping information that P1 can now use to update his knowledge.

The basic problem connected with the approach outlined above is the integration of newly acquired conceptual knowledge of other agents into the already existing background knowledge. Just adding the corresponding statement to the knowledge base will often lead to conflicts that have to be resolved. Techniques of relief revision of defeasible reasoning might provide solutions for this problem.

## **Conclusions**

We argued that query processing is an important methods on the semantic web and is likely to gain the same central role it has in databases and information systems. We showed that query processing in the setting of the semantic web poses new challenges that raise many well-known problems of Artificial Intelligence research such as reasoning, integration, approximation, transformation and discovery. We outlined the problems connected with these requirements on a rather informal level and pointed to first results on some subproblems.

The main message of this paper is that there is a need for intensive research in the areas mentioned in order to facilitate query answering. Maybe the most central question is how to combine well-founded logical methods of reasoning about information with background knowledge with heuristic and approximate methods. While the first are desirable from a theoretical and conceptual point of view, the latter are necessary in order to cope with the lack of consistency, completeness and the need for scalability. The success of every approach will rely on finding the right balance between formal rigor and powerful heuristics.

## References

Dan Brickley, R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema *W3C Working Draft* 30 April 2002 This Version: <http://www.w3.org/TR/2002/WD-rdf-schema-20020430/>

Jerome Euzenat and Heiner Stuckenschmidt. The 'Family of Languages' Approach to Semantic Interoperability. In B. Omelanyenko and M. Klein. *Knowledge Transformation for the Semantic Web*. IOS Press, to appear.

Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D. L., & Patel-Schneider, P. F. (2001). OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38--44.

Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer D. Widom. *Database Systems: The Complete Book*. Prentice-Hall 2002.

Asuncion Gomez-Prez, Oscar Corcho, *Ontology Languages for the Semantic Web*, *IEEE Intelligent Systems*, Volume 17, Issue 1. January/February 2002.

Alon Y. Halevy. Answering queries using views: A survey *Volume 10 , Issue 4 , pp 270 -294 The VLDB Journal 2001*

Ian Horrocks and Sergio Tessaris. Querying the semantic web: a formal approach. In Ian Horrocks and James Hendler, editors, *Proc. of the 2002 International Semantic Web Conference (ISWC 2002)*, number 2342 in Lecture Notes in Computer Science. Springer-Verlag, 2002.

Ian Horrocks and Sergio Tessaris. A conjunctive query language for description logic aboxes. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI-2000)*, pages 399-404, 2000.

Deborah L. McGuinness, Frank van Harmelen *Feature Synopsis for OWL Lite and OWL W3C Working Draft* 29 July 2002 <http://www.w3.org/TR/2002/WD-owl-features-20020729/>

Heiner Stuckenschmidt. Approximate Information Filtering with Multiple Classification Hierarchies. *Special Issue on Internet Intelligent Systems of the International Journal of Computational Intelligence Applications*, to appear 2002.

Heiner Stuckenschmidt, Frank van Harmelen and Fausto Giunchiglia. Query Processing in Ontology-Based Peer-to-peer Systems. Technical Report, *AI Department Vrije Universiteit Amsterdam*, November 2002.

Heiner Stuckenschmidt and Frank van Harmelen. Approximating Terminological Queries. In T. Andreasen et.al. *Flexible Query Answering Systems*, Lecture notes in AI, Springer 2002.