A Model-Driven Approach for Designing Distributed Web Information Systems

Richard Vdovjak* and Geert-Jan Houben

Eindhoven University of Technology,
POBox 513, 5600MB, Eindhoven,
The Netherlands
{r.vdovjak,g.j.houben}@tue.nl

Abstract. There is an apparent need for specifying the integration of multiple knowledge sources during the design of Web Information Systems (WIS) where the actual data is often retrieved from several content providers. Despite that, there exists very little work on integration within the context of WIS engineering and the related design methodologies in particular. We argue that this new context brings several additional requirements which must be dealt with in order to be able to successfully deploy distributed WIS. In this paper we elaborate a model that covers the integration phase of the WIS design trajectory. We centered our approach around the emerging data standard on the Semantic Web – RDF. The proposed integration model is able to reconcile many semantic heterogeneities that frequently occur among disparate RDF sources. We also address the issues of distributed RDF query processing and optimization, and test the performance of our framework.

1 Introduction

The need for handling multiple sources of knowledge and information is very apparent in the context of engineering Web Information Systems (WIS). The actual data presented in a typical WIS is often retrieved from several (possibly heterogeneous) set of sources. While the integration problem was carefully studied in isolation in the database field, there exists very little work on integration in the context of Web engineering applications and of the design methodologies that support them in particular. We argue that this new context brings several additional requirements that must be dealt with in order to be able to successfully design a distributed WIS. The separation-of-concerns principle together with the model-based approach has proven to be an efficient remedy to the complexity of the WIS design. In this paper we elaborate a model that covers the integration and data retrieval phase of the WIS design trajectory. We centered our approach around the emerging data standard on the Semantic Web, the Resource Description Framework (RDF) [1]. RDF is the modeling foundation of the Semantic Web. Despite its inherently distributed nature, most of the current RDF processing engines store the RDF data locally as a single knowledge

* Richard Vdovjak is also affiliated with Philips Research Eindhoven, The Netherlands

D. Lowe and M. Gaedke (Eds.): ICWE 2005, LNCS 3579, pp. 453–464, 2005.

repository, i.e. RDF models from remote sources are replicated and merged into a single model. Distribution is retained only virtually through the use of namespaces to distinguish between different models. We argue that many interesting applications on the Semantic Web would benefit from, or even require an RDF infrastructure that supports real distribution of information sources that can be accessed from a single point. In this paper we focuss on the RDF integration problem taking into account the context of WIS.

The rest of the paper is structured as follows. In section 2 we introduce the Hera WIS design framework and derive some WIS specific requirements for the integration phase. Section 3 summarizes the semantics of RDF(S) and formally defines some important terms used in our integration suite. Section 4 describes the Integration Model formalism which is able to deal with many semantic heterogeneities that frequently occur among sources on the Semantic Web. Section 5 addresses the issues of distributed query processing and optimization, and describes the performance of our system. Section 6 presents concluding remarks.

2 Modeling WIS in Hera

A primary focus of the Hera project is to support Web-based information system (WIS) design and implementation. A WIS generates a hypermedia presentation for the data that is retrieved from the data storage in response to a user query. This entire process of retrieving data and presenting it in hypermedia format needs to be specified during the design of the WIS. The typical structure of the WIS design in the Hera perspective consists of three layers: the semantic layer focusing on the application's semantics and the integration aspects, the application layer designing a navigational view over the data, and the presentation layer dealing with a concrete rendering platform such as HTML, WML or SMIL.

In this paper we focus mainly on the semantic layer of the Hera methodology. After the process of integration, the conceptual model instances are generated as response to a user query. Figure 1 presents an overview of the semantic layer with its central component – the mediator.

2.1 Related Work

As opposed to Hera, most of the web engineering approaches (e.g. UWE [2], WebML [3] or XWMF [4]) do not explicitly consider integration. A notable exception is the OntoWebber [5] system which we detail below.

OntoWebber is a system for building and managing data-intensive websites. Similarly to Hera, it adopts a model-driven ontology-based approach for declarative website management and data integration. It advocates the use of ontologies as the basis for constructing different models necessary for WIS design. OntoWebber supports the integration of heterogeneous data sources based on RDF as common format for modeling semistructured data. In the first step OntoWebber focuses on syntax reconciliation converting all source data into RDF. This RDF data (both the schema and the instances) is replicated and stored locally. From

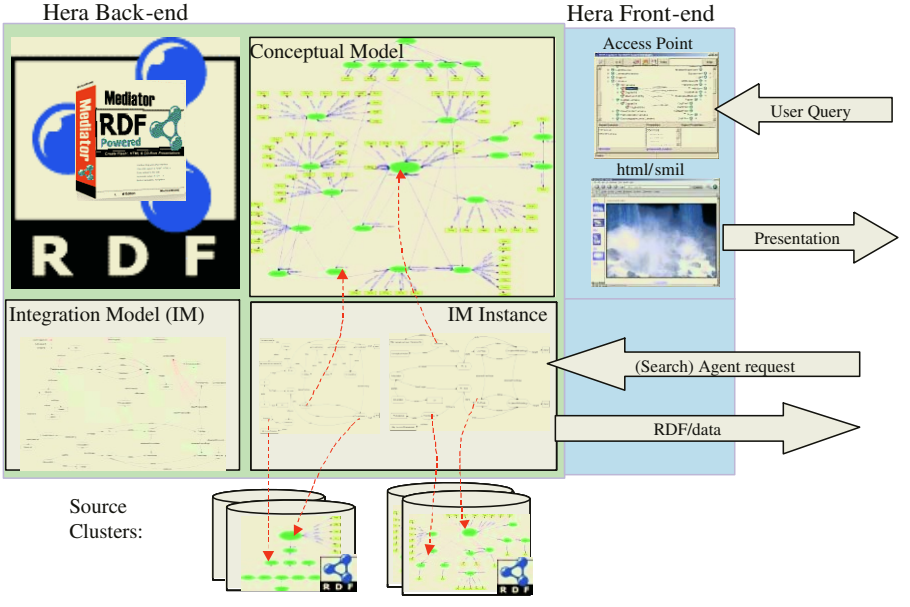

Fig. 1. An overview of the semantic layer of Hera

this point of view, OntoWebber acts as a data warehouse and does not guarantee the freshness of its data, as opposed to the Hera integration framework which implements the on-demand retrieval paradigm, assuring that the retrieved data is always up-to-date. After the replication phase, the local copies of the source data are articulated by the designer in terms of a reference ontology which captures the domain of interest.

2.2 Hera Requirements for RDF(S) Data Integration in the Context of WIS

By analyzing the specifics of WIS, we form the following set of requirements for our integration framework. The existing approaches to data integration, e.g. [6] often do not meet some of these important prerequisites which renders them less suitable for designing WIS. Below we summarize the list of requirements for the Hera integration framework.

– The use of (open) WWW standards.
While in the past the main consumer of the information provided by WIS was usually a human, currently there are more and more (Web) applications that need to process this information as well. Moreover, WIS themselves often consist of a complex composition of collaborating (Web) components, such as the components in the Hera software suite. This requires the use of Web standards throughout the entire process. To promote the re-use of integration specifications by other parties on the Semantic Web, it is useful

that both the information to be integrated, and the integration specification itself are expressed in the same description standard.
- Ontology level, model-based approach.
  Since WIS are data-intensive applications with large numbers of instances per concept, it is not feasible to specify integration mappings for every individual instance. This data-intensive nature combined with the separation-of-concerns principle adopted by Hera implies that the integration should be expressible in an explicit model which reasons in terms of source ontologies rather than in terms of the actual data instances. The instance integration has to be realized on-the-fly during the query evaluation.
- Expressivity of the integration formalism.
  The integration formalism must be able to cover a wide range of semantic heterogeneities frequently occurring on the Semantic Web. This includes schema heterogeneities where the source ontologies can differ in their concepts, properties, and structure. In particular, RDFS models often contain sequences of connected properties – so-called ontology or schema paths. These paths can differ considerably among the sources and the conceptual model. To reconcile these discrepancies is an essential prerequisite to successful evaluation of join queries across multiple sources.
- Freshness of data.
  Hera has the ambition to support the engineering of WIS which often use as content providers other autonomous Web sources. Both the structure and the data of these sources may change without notification. While the structural changes are assumed to be less frequent, the actual data can change frequently. In this context the freshness of the gathered data is an important aspect and should be guaranteed.

3 RDF(S) and Ontologies

An RDF triple model is similar to a directed labeled graph [1]. The nodes in the graph are used to represent resources or literals. Literals (strings) denote content that is not processed further by the RDF processor[1]. The nodes that represent resources can be further classified as nodes representing URI references or blank nodes. All non-blank nodes are (explicitly) labeled with resource identifiers (URIs) or string values. The edges in the graph represent properties and are also labeled by URIs. The set of all labels occurring in the graph is called the vocabulary of the graph. In order to associate formal semantics to an RDF graph, all labels in its vocabulary must have an interpretation. An interpretation specifies for every URI reference what it stands for as well as whether it is a property, resource, or a literal value. In case of a property it also describes what value that property can take for things in the domain of discourse. To be able to reason about sources at schema level we must extend the initial vocabulary

[1] In this work we consider only plain literals but our approach can be easily extended to typed literals as well

with the standardized set of URI references defined in the RDF vocabulary and RDF Schema vocabulary [7].

Definition 1 *An RDFS interpretation of a vocabulary V is defined as 7-tuple:*

$$(IR, IP, IC, LV, IS, IEXT, ICEXT)$$

where IR is a set of resources, IP is a set of properties, IC is a set of classes, LV is a set of literal values, IS is a mapping $V \rightarrow IR$ defining the interpretation (meaning) of URI references and literals, and $IEXT$ is a mapping $IP \rightarrow 2^{IR \times IR}$ defining the extent of properties[2]. $ICEXT$ defines an extent of every class. Moreover, every RDFS interpretation has to satisfy several semantic conditions, e.g. the transitivity and reflexivity of $rdfs{:}subClassOf$ and $rdfs{:}subPropertyOf$. We refer to [7] for the complete condition list.

3.1 RDFS Ontologies

Both, the WIS that we need to populate with data and the sources are represented by their ontological descriptions. The ontology captures their domain of discourse. When we refer to ontology, we mainly mean the hierarchy of concepts together with their properties. The actual data that populates these concepts is referred to as ontology instances. Note that due to the intensional nature of RDF(S) semantics the antisymmetry property is not guaranteed to hold and therefore we cannot say that there is a partial order on the set of classes and properties. In the ontology definition below we use the pre-order relation – a binary relation that satisfies the reflexivity and transitivity, but not necessarily the antisymmetry.

Definition 2 *An RDFS ontology O as a 4-tuple $(G, I, \preccurlyeq_{IC}, \preccurlyeq_{IP})$, where*

- *G is an RDF graph.*
- *I is an RDFS interpretation of the vocabulary of G and I holds that $IR = IP \cup IC$, i.e. there are no 'instance' resources.*
- *$\preccurlyeq_{IC}$ is a pre-order relation on the set IC defined by $IEXT(I(rdfs{:}subClassOf))$*
- *$\preccurlyeq_{IP}$ is a pre-order relation on the set IP defined by $IEXT(I(rdfs{:}subPropertyOf))$*

If an RDF graph (and its interpretation) adheres to a given ontology O, i.e. uses the properties and classes defined in O, it is considered to be an instantiation of O.

Definition 3 *Let G be an RDF graph and I its RDFS interpretation. The pair (G, I) is an instantiation of the ontology O $(G', I', \preccurlyeq_{IC}, \preccurlyeq_{IP})$ if the following holds: $\{c \mid (x, c) \in IEXT(I(rdf{:}type))\} \subset I'.IC$, and $I.IP \subset I'.IP$, and the types from the extent of every non-system property in G follow the types defined in O by the $I'(rdfs{:}domain)$, and $I'(rdfs{:}range)$ for that particular property.*

[2] When an interpretation I is applied to a single URI reference, it represents a straightforward application of the IS function of I

Definition 4 *A single path in ontology* O $(G, I, \preccurlyeq_{IC}, \preccurlyeq_{IP})$ *is defined as triple* (C, P, T) *where* $C \in I.IC$, $P \in I.IP$, $T \in I.IC \cup \{I(rdfs{:}Literal)\}$ *and* $(P, C) \in IEXT(I(rdfs{:}domain))$ *and* $(P, T) \in IEXT(I(rdfs{:}ramnge))$.

Informally, a single ontology path is a property together with its domain and range classes. By using the following definition we can combine single ontology paths into an arbitrarily long path expression.

Definition 5 *Let* $p_1, \ldots p_n$ *be a sequence of single paths. They together constitute a composed ontology path expression if for any neighboring single paths* $p_i(C_i, P_i, T_i)$ *and* $p_{i+1}(C_{i+1}, P_{i+1}, T_{i+1})$ *the following holds:*

$$\begin{gathered} T_i \neq I(rdfs{:}Literal) \wedge \\ ((P_i, C_{i+1}) \in IEXT(I(rdfs{:}range)) \vee \\ (P_{i+1}, T_i) \in IEXT(I(rdfs{:}domain)) \vee \\ (T_i, C_{i+1}) \in IEXT(I(rdfs{:}subClassOf)) \vee \\ (C_{i+1}, T_i) \in IEXT(I(rdfs{:}subClassOf))) \end{gathered}$$

4 Integration Model

The official RDF semantics [7] makes a strong assumption about URI references: they are assumed to be globally coherent, so that a single URI reference can be considered to have the same meaning wherever it occurs. However, in the heterogenous world of the WWW we often have to relax this assumption if we want to integrate data from different sources. In Fig. 2 we depict an example of two sources with different vocabularies and interpretations. Both however describe the same domain and the integration model is a way to specify how they relate to the conceptual model. The solid lines indicate mappings established during the integration phase, and the dashed lines denote results during the query answering process[3]. The integration model defines a set of articulations which create semantic mappings between the sources and the CM.

4.1 Articulations

The simplest form of articulation is that combining two single path expressions. Informally, this articulation establishes a schema level link between two properties (one from a source and one from the CM). During the query resolution, the instances of the source property are translated into instances of the property from the CM.

Definition 6 *Let* O_S *be an RDFS ontology describing the source* S *and* O_{CM} *the (traget) RDFS ontology describing the WIS which we need to populate with instances. A simple articulation As is defined as a pair* (Q, R)*, where* $Q(C, P, T)$ *and* $R(C, P, T)$ *are single ontology paths in* O_{CM} *and* O_S*, respectively. At instance level, the As is a function which transforms the source instances into CM*

[3] The property extent mappings $IEXT$ are omitted for the sake of readability

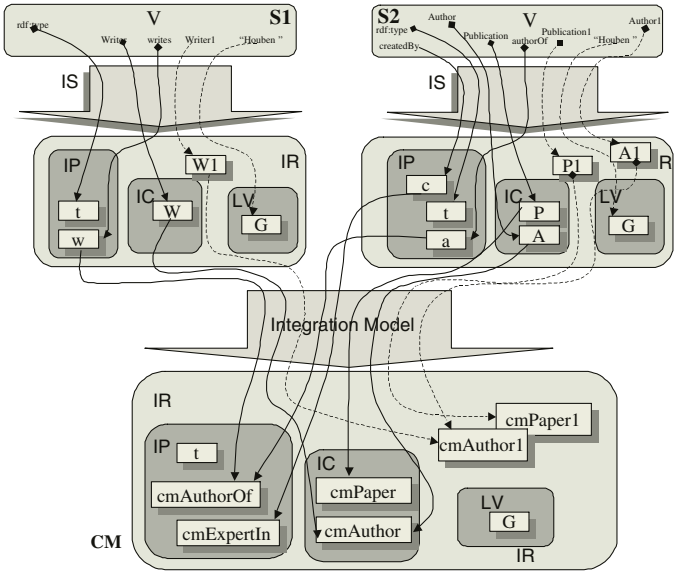

Fig. 2. Sources with different vocabularies and interpretations linked to the CM

instances. Let $Inst_S$ be the instantiation of the source ontology O_S. The instantiation of the O_{CM} is augmented such that the following holds:
$\forall(x,y) \in Inst_S.I.IEXT(R.P)\ \exists(x',y') \in Inst_{CM}.I.IEXT(Q.P) \wedge (x', Q.C) \in Inst_{CM}.I.IEXT(I(rdf{:}type)) \wedge ((y', Q.T) \in Inst_{CM}.I.IEXT(I(rdf{:}type)) \vee Q.T = I(rdfs{:}Literal))$.

Although a simple articulation allows for different vocabularies, when also the structure of the two ontologies differs we need a more sophisticated integration tool. A *path articulation* defined below allows for the mapping of ontology path expressions of different length. The idea is to link the beginning and the end of the two paths and to apply simple articulations where they exist. In case there exist some parts of the path in the CM which are not covered by the source path expression, e.g. due to structural heterogeneity, we generate blank node instances in the CM to keep the path connected.

Definition 7 *Let O_S be an RDFS ontology describing the source S and O_{CM} the (traget) RDFS ontology describing the WIS which we need to populate with instances. A path articulation Ap is defined as a pair (p,q) where p, and q are ontology path expressions in O_{CM} and O_S, respectively.*

Let (C_{1x}, P_{1x}, T_{1x}) and (C_x, P_x, T_x) denote respectively the first and the last single path of the composed path expression x. The semantics of Ap is defined as the augmentation of the instantiation of the O_{CM} in the following way. For every instance of the path q we assume an instance of the path p consisting of blank nodes between the defined properties of p such that the path is connected. We generate instances to replace (some of) the blank nodes such that the fol-

lowing holds: $\forall x \in Inst_S.I.ICEXT(C_1q)\ \exists x' \in Inst_{CM}.I.ICEXT(C_1p)$ *i.e. we generate class instances for the beginning of the path p. We do the same also for its end:* $\forall x \in Inst_S.I.ICEXT(T_1q)\ \exists x' \in Inst_{CM}.I.ICEXT(T_1p)$. *Unless the end is a literal property, i.e.* $T\_S = I(rdfs{:}Literal)$, *then the literal values are copied:* $\forall (x,y) \in Inst_S.I.IEXT(P\_q)\ \exists (x',y') \in Inst_{CM}.I.IEXT(P\_p) : y = y'$. *Further, we generate appropriate instances and replace the blank nodes for every single path in p that has a simple articulation that associates it to a single path from q.*

To cover also the cases when a (literal) value in the conceptual model is obtained from several paths, possibly distributed among different sources, we introduce a multiple source articulation. A multiple source articulation is defined as a tuple $(p_T, q_{S_1}, \ldots q_{S_n}, Concat)$, where p_T is an ontology path expression in O_{CM}, $q_{S_1}, \ldots q_{S_n}$ are ontology path expressions in source otologies, and $Concat$ is a function defining the concatenation result. Note that this articulation applies only for literal values. We omit the description of its semantics as it is a straightforward extension of the path articulation. For the converse where several values in the CM are obtained by splitting one value from a source we have a multiple CM articulation: $(p_{CM_1}, \ldots p_{CM_n}, q_S, Split)$.

4.2 RDF(S) Representation of the Integration Model

As we stated in the requirements, it is useful that the integration framework can be expressed in the same data format as the actual data that are integrated. It both facilitates the semantic interoperability and allows for reasoning about the integration phase at a higher level of abstraction. We translated the concepts of our theoretical integration framework into an RDF schema called Integration Model Ontology (IMO). This ontology describes in the RDF syntax the notion of path expressions, articulations, etc. The integration model instances (IMI) are created by the designer (or generated by a mapping tool) for a concrete integration problem. Due to lack of space we do not present in detail the verbose RDF serialization here, interested reader is referred to the Hera website[4]. The IMO together with IMI are used by the mediator during the query processing.

5 Distributed RDF Query Processing

The mediator component is responsible for finding the answer to the user query by consulting the available sources based on the integration model instances. The mediator takes as input the user query formulated in SeRQL [8].

```
SELECT A, P
FROM  {A} expertIn  {TA};  authorOf  {P},
      {P} concerns  {TP};  frontPage {F},
      {F} mentions  {A}
WHERE TA=TP
```

[4] http://wwwis.win.tue.nl:8080/~ hera

To illustrate the query processing in our mediator, consider the above SeRQL query example[5]. In this example we assume a total distribution, i.e. all properties reside on different sources and the mediator performs all partial joins.

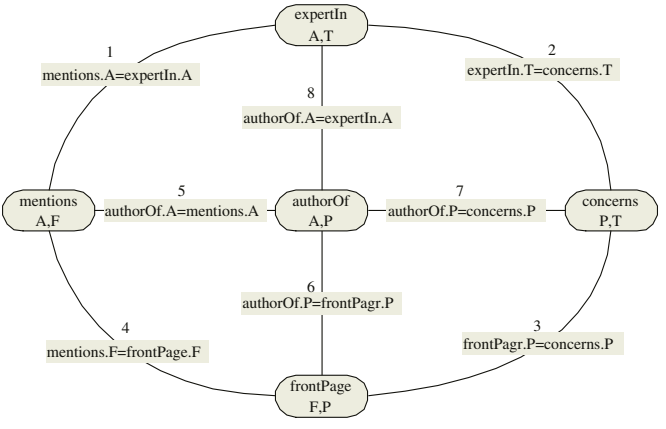

Fig. 3. Join graph

The query in our example is processed as follows.

- We first normalize the SeRQL query in the sense that all variables that are explicitly declared equal in the *Where* clause are given identical names (variables TP and TA are both renamed to T).
- From the normalized query we create a join graph (see Fig. 3) in which every property in the *From* clause stands for a relation with two attributes defined by the variable names. In the join graph we represent these relations as nodes and connect every two nodes that share an attribute[6]. Every edge in the join graph is interpreted as a join condition between the two connected relations.
- Next, we assign a random order to the edges of the join graph (the order is depicted by numbers above the edges) and recursively fold the join graph by combing the two nodes that are connected by the edge with the smallest number. The folding sequence for our join graph is depicted in Fig. 4. Dashed lines indicate the edges which are removed in that particular folding step[7]. The folding sequence essentially represents a query plan: a folding step is equivalent to a join operator and every edge which is removed in that particular step represents one conjunct in the join condition. To minimize the execution costs, this initial query plan is subsequently optimized by reordering the joins.

[5] *"Retrieve all authors and their papers, where the author is an expert in the topic that concerns the paper and he is also mentioned on the front-page of the paper"*

[6] If the nodes share more attributes, we create one edge for each such attribute

[7] For the sake of clarity, the node names were abbreviated to their starting letters

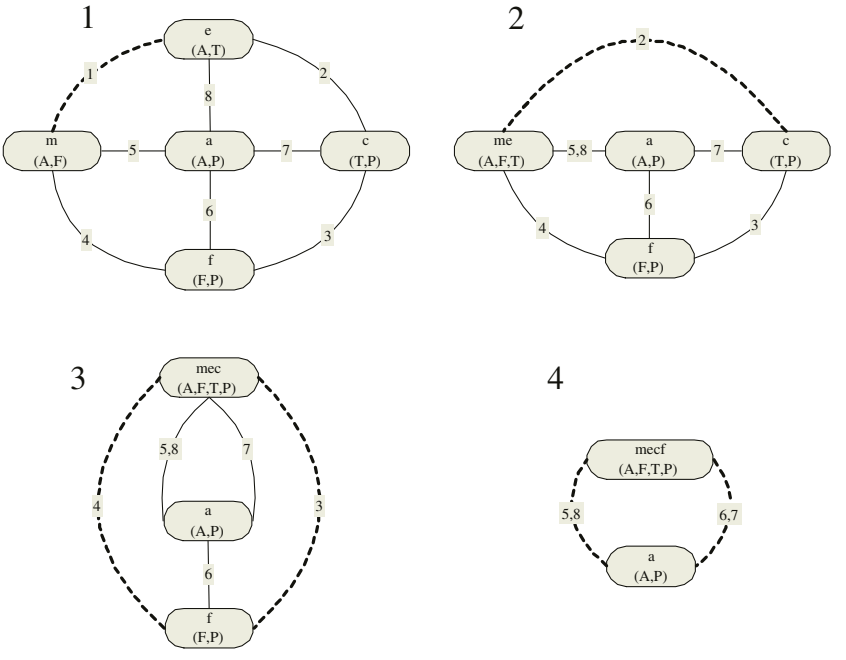

Fig. 4. Join graph folding

– To consult the sources, the mediator locates for every property in the join graph an articulation. From this articulation, it determines the source that provides an answer to it together with an appropriately translated path expression. This path expression is transformed into a SeRQL query and executed on that particular source. The sources are consulted in a multi-threaded fashion in order to achieve a high degree of parallelization. The mediator subsequently collects the partial results and performs the necessary join operations according to the query plan determined by the optimizer.

5.1 Query Optimization and Performance Evaluation

To test our integration framework we synthesized an RDFS schema of approximately 50MB and instantiated it with 500MB of RDF instances. Note that a schema/instance ratio of 10% is quite large; in normal circumstances, the size of the schema seldom reaches even 1% of the size of the instances. This represented for us a worst case scenario since the mediator has to join partial path results, the bigger schema the more potential paths to join. The data set was distributed among several computers connected by the Internet, and the underlying sources were using the Sesame RDF storage system[8].

Note that the sources, unlike the mediator, contain instance indices and are therefore very efficient for joining the path queries. Creating an instance index at

[8] http://www.openrdf.org/

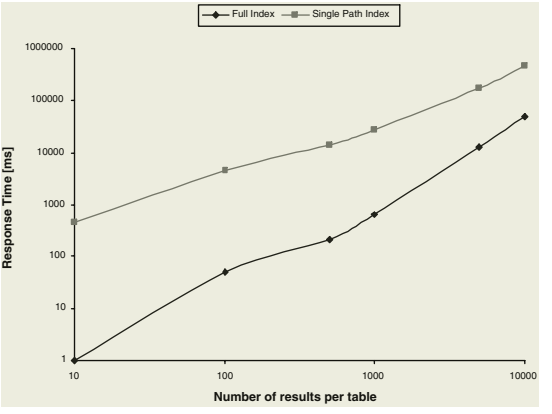

Fig. 5. The improvement with the use of path indexing

the mediator would require to gather all the data from the sources and thus turning our virtual repository into a datawarehouse with all its disadvantages like the freshness problem, the maintenance problem, data ownership issues etc. Instead, we adopted a schema based approach. Since the mediator has the schemas of the underlying sources, in order to minimize its workload a sophisticated ontology path indexing takes place. The main idea is to push down to the sources the longest possible paths they can answer, reducing the number of joins at the mediator. The performance improvement gained by schema indexing is depicted in Fig. 5.

While the path indexing is clearly beneficial, especially for larger results sets the joining at the mediator still represents a bottleneck. In order to minimize the joining time, which in turn due to different cardinalities and join selectivities largely depends on the order in which the joins are performed, we implemented a join ordering heuristic as a combination of iterative improvement and simulated annealing [9]. As depicted in Fig. 6, this improved the performance of the system even further, especially after the initial calibration phase.

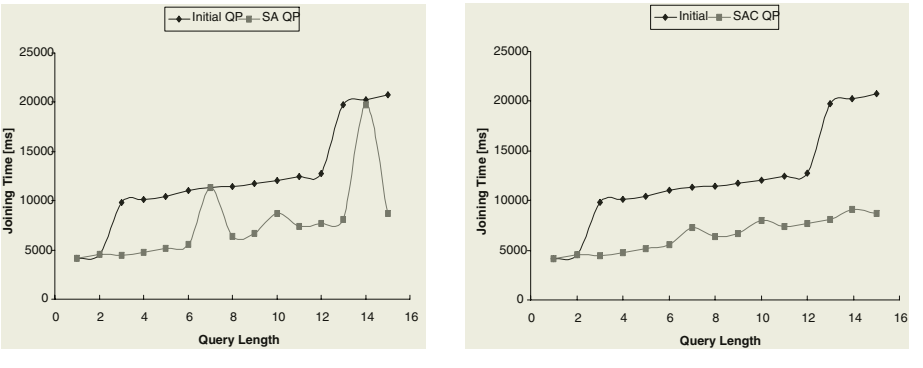

Fig. 6. The improvement with the IISA, left: uncalibrated, right: calibrated

6 Conclusions

As the nature of WIS changes under the influence of the Semantic Web (SW) initiative, the need to capture the semantics of the application data increases. In the typical WIS, the data is often gathered from several sources and it is crucial to understand their semantic annotations. We derived a set of integration requirements in the context of WIS. To address them, we presented our integration framework that helps to overcome semantic heterogeneities of RDFS meta-data from different sources. We also evaluated the performance of our system and proposed several optimization techniques to improve the the query evaluation. As future work we intend to extend the expressive power of our integration model towards higher level ontology languages such as OWL, and to investigate other possibilities to improve the performance of our implementation. One of the promising directions to minimize both the mediator's workload and the data transfer from the sources is to establish a network of collaborating mediators that would perform some query processing tasks, e.g. joins, on request.

References

1. Ora Lassila and Ralph R. Swick. Resource description framework (rdf) model and syntax specification. W3C Recommendation 22 February 1999.
2. Nora Koch, Andreas Kraus, and Rolf Hennicker. The authoring process of the uml-based web engineering approach. In *First International Workshop on Web-Oriented Software Technology*, 2001.
3. Stefano Ceri, Piero Fraternali, and Maristella Matera. Conceptual modeling of data-intensive web applications. *IEEE Internet Computing*, 6(4):20–30, 2002.
4. Reinhold Klapsing and Gustaf Neumann. Applying the resource description framework to web engineering. In *Electronic Commerce and Web Technologies, First International Conference, EC-Web 2000*, volume 1875 of *Lecture Notes in Computer Science*, pages 229–238. Springer, 2000.
5. Yuhui Jin, Stefan Decker, and Gio Wiederhold. Ontowebber: A novel approach for managing data on the web. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, pages 488–489, 2002.
6. Jeffrey D. Ulman. Information integration using logical views. In *Proceedings of the 6th Int. Conference on Database Theory, ICDT'97*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1997.
7. Patrick Hayes. Rdf semantics. W3C Recommendation 10 February 2004, 2004.
8. Jeen Broekstra and Arjohn Kampman. Query language definition, (on-to-knowledge eu-ist-1999-10132 deliverable 9). Technical report, Aidministrator Nederland b.v., 2001.
9. M. Steinbrunn, G. Moerkotte, and A. Kemper. Heuristic and randomized optimization for join ordering problem. *The VLDB Journal*, 6:191–208, 1997.