4

Integrating Disparate Digital Libraries using the WASSIT Mediation Framework

Faouzia Wadjinny, Imane Zaoui, Ahmed Moujane and Dalila Chiadmi
Computer Sciences Department, Mohammadia Engineering School, BP 765
Rabat Agdal,
Morocco

1. Introduction

Nowadays, there is a trend to integrate several digital libraries (DLs) to offer richer information. However, the following three characteristics of DLs make their integration a difficult task (Hasselbring, 2000): (i) Distribution: geographical spread; (ii) Heterogeneity: difference at both the technical level (e.g., hardware platform, operating system, etc.) and conceptual level (e.g., data model, query language, etc.); (iii) Autonomy: DLs are self-sufficient, as opposed to being delegated a role only as components in a larger system. Therefore, challenges faced when integrating DLs include interoperability (among different DLs) and resource discovery (selection of the best sites to be integrated). There are two different types of interoperability for DLs integration (Shen, 2006): syntactic interoperability and semantic interoperability. Syntactic interoperability is the application-level interoperability that allows multiple software components to cooperate even though their data model, query language, interfaces, etc. are different. Semantic interoperability is the knowledge-level interoperability that allows digital libraries to be integrated, with the ability to bridge semantic conflicts arising from differences in implicit meanings, perspectives and assumptions, thus creating a semantically compatible information environment based on agreed-upon concepts.

To deal with the interoperability problem, two solutions can be used: warehousing and mediation systems. In the warehouse approach (Rundensteiner and al., 2000), information is in some way periodically extracted from different sources, processed, merged with information from other sources, and then loaded into a centralized data store. Queries are posed against the local data without further interaction with the original sources. Modifications are filtered (e.g. for relevance or update-time) and propagated in some manner to upgrade the data warehouse. The main advantage of the warehousing approach is the performance of query processing. The main drawbacks are that the data may not be fresh and adding new data source requires reconsidering the warehouse schema. Thus, concerns about data quality and consistency must be addressed.

In mediation systems (Wiederhold, 1992), data remains at the sources and queries to the integrated system need to be translated, at run time, into a sequence of sub-queries to the underlying data sources. Data is not replicated and is guaranteed to be fresh at query time. However, a considerable performance penalty must be paid because sources are contacted for every query. Besides, in heterogeneous environments, especially in the context of DLs,

sources may have diverse and limited query capabilities. Thus, not all of the translations are feasible. Therefore, another challenge faced when integrating DLs is how to generate efficient and feasible query plans to retrieve data from DLs.
Our solution for integrating disparate DLs is a mediation framework, called WASSIT *(frame**W**ork d'intégr**A**tion de re**SS**ources par la méd**I**a**T**ion)*. In this chapter, we describe the features of WASSIT. In particular, we present how DLs are selected and ranked according to the user quality requirements. Since syntactic interoperability is treated implicitly in mediation systems (by using a common data model and wrappers), we will focus on our solution for semantic interoperability. Generating feasible and efficient query plans, by WASSIT, is also part of this chapter.

Chapter overview

In Section 2, we describe the high level architecture of our mediation framework WASSIT. We present how WASSIT selects pertinent DLs that satisfy the user quality preferences in section 3. In section 4, we present our solution for semantic interoperability. Our approach for optimizing queries over DLs with limited capabilities is presented in section 5. Performances evaluation is discussed in section 6. We conclude and present our future works in section 7.

2. High-level architecture of WASSIT

Our mediation framework WASSIT relies on the well-known mediator architecture (Wiederhold, 1992). WASSIT defines an infrastructure which provides the generic structure and the behaviour of a set of reusable components in an information mediation context (Zellou, 2008). Our framework is mainly made up of two principal components: Mediator and Wrappers. The Mediator, which is the query processing core of the framework WASSIT, has to decompose a user query into a set of sub-queries targeted to the sources. Each sub-query is transmitted to the corresponding source via the associated wrapper. The answers delivered by the wrappers are then combined to form the response to the initial query. The high level architecture of WASSIT is shown in figure 1. In this architecture, we distinguish three levels: the source level including the data sources and the wrappers, the mediation level containing the mediator, and finally the user level containing the user interface. In the mediation level, WASSIT is composed of six modules and a knowledge base.

2.1 Our technological choices for WASSIT

DLs generally differ with respect to the structures they use to represent data (e.g. tables, objects, files, and so on). We use XML as a common data model in WASSIT to reconcile sources' heterogeneous data models because it provides a common format for expressing both data structures and contents. Thus, it can integrate structured, semi-structured and unstructured data. XQuery (Fernández and al., 2007) is the query language we adopt in WASSIT since it is the W3C standard for querying XML documents. In order to achieve efficient query processing, we represent the queries according to an algebraic model. The one we have chosen is XAT (Wadjinny & Chiadmi, 2006). XAT algebra offers SQL operators such as union, join, etc. It offers also specific operators such as navigate, tagger, etc. Ontologies in WASSIT are used at two levels: to represent schema mappings and to capture the semantic of each source since we address semantic heterogeneity. We adopt OWL (Deborah and al., 2004), the Ontology Web Language, to represent these ontologies.

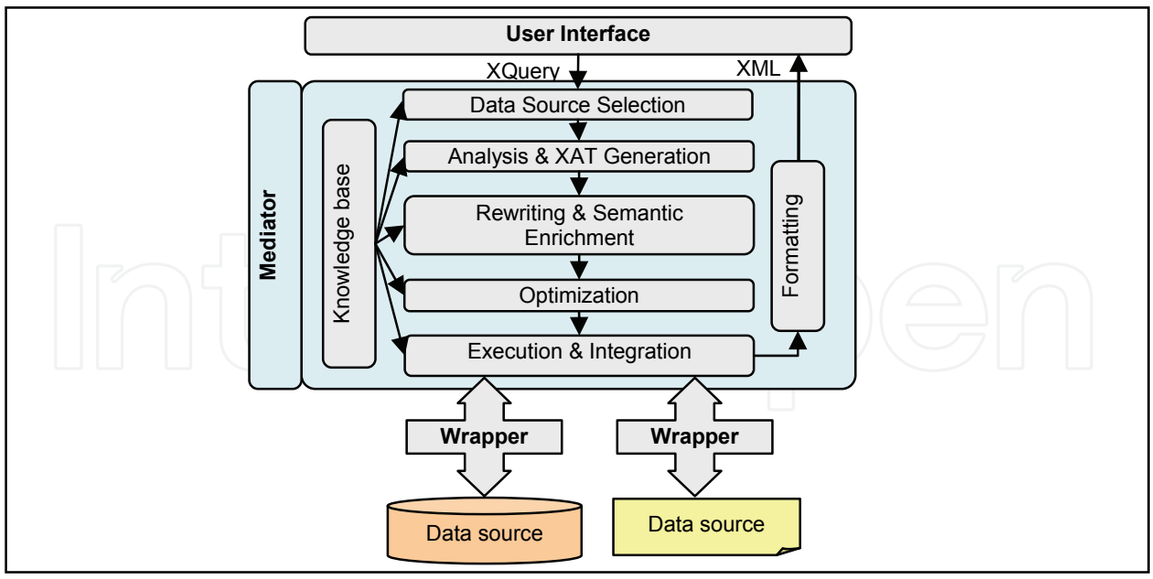


Fig. 1. High level architecture of WASSIT

2.2 Description of WASSIT's components

As cited earlier in this section, WASSIT is made up of a user interface, a mediator which contains six modules and a knowledge base, and wrappers. In the following, we will present each entity of WASSIT.

User Interface*:* It is a QBE (Query By Example) interface which frees the user from the knowledge of the XQuery language. In addition, this interface allows users to formulate their queries using the concepts of the global ontology. After the reception of a query by this interface, the corresponding XQuery query is generated. Moreover, the user interface allows users to express their preferences and needs through a user profile.

Data Source Selection **module**: To select the most relevant sources, this module performs quality matching between a user's profile and sources' profiles. The selected sources are then integrated to get a personalized response that respects user's quality requirements. More details about this module are given in section 3.

Analysis & XAT Generation **module:** This module has to analyze the user's queries. It rejects the syntactically incorrect ones. It eliminates also the queries that refer to unavailable concepts. This is achieved by using the knowledge base. User's queries are then transformed into XML algebra trees in order to be treated. Each node of a XAT tree is an algebraic operator.

Rewriting & Semantic Enrichment **module:** Let's remind that our framework aims to access a set of heterogeneous information sources. Every source has its local schema that describes its structure in a data model. The query submitted to the framework is formulated in terms of mediated schema (global schema). To have the query executed, the framework must rewrite the user's query formulated in terms of mediated schema as a query execution plan (QEP). Each QEP is presented in the form of a tree, where leafs are sub-queries that will be sent to the wrappers, and nodes are reconstruction operators that will be used by the mediator to integrate the results. The *Rewriting* module generates a QEP through three steps; each step is processed by one sub-module (Gounbark and al., 2009). These sub-modules are described in the following.

(*1*) *Global views substitution* module*:* This module has two features. First it ensures global views substitution, which consists in replacing each global view reference by the definition of this view. Views definitions are retrieved from the mapping definition. Then global paths (used to define global query) are projected on local paths (used to define local views). (*2*) *Union and join Operator ascending* module*:* Join and union operators, having distinct views from distinct data sources, can't be executed by a source. Thus, these operators have to be executed by the mediator. In order to schedule their execution, they are moved at the top of the algebra tree. (*3*) *Bindings adjustment* module: moving operators across the plan tree (previous step) makes parameters inappropriate. Consequently, this module has to adjust binding operators' parameters.
Moreover, in this step, we enrich semantically each sub-query (when possible) with synonyms, hyperonyms and hyponyms.
***Optimization* module:** The *Optimization* module takes as input the QEP obtained after query rewriting. After extracting sub-queries from this QEP, the *Optimization* module constructs a plan according to query capabilities of the underlying DLs. More details about this module are given in section 5.
***Execution & integration* module:** This module takes as input the sub-queries delivered by the *Optimisation* module and sends them to the appropriate wrappers using the localization information given by the knowledge base. It is composed by three sub-modules (Gounbark and al., 2009), which are described in the following.
(*1*) *XQuery Query Generator* module: sub-queries are represented by a XAT tree. This module translates each XAT tree sub-query to a XQuery query. (2) *Sub-queries Execution* module: this module ensures the actual execution of XQuery queries. It sends the XQuery queries to the right wrapper which translates the XQuery query to the underlying data source querying language. (*3*) *XAT Table Generation* module: This module constructs a XAT table from the XML results returned by the wrappers. The resulting XAT tables are combined according to the optimized QEP to form the answer to the user query.
***Formatting* module:** In this module, the result returned by the *Execution & Integration* module is formatted in order to form the answer which will finally be returned to the end user.
The knowledge base: The knowledge base is associated to the mediator, it stocks the general information used for query processing in the framework. It contains global ontology, local ontologies, users' profiles, sources' profiles, physical localization of sources, source descriptions, localization of wrappers, source capabilities, etc. More details about the construction of global ontology are given in section 4.
Wrappers: At a given wrapper, a sub-query expressed in XQuery is translated into the source query language. The wrapper has also to format the results returned by the source in an XML format. In WASSIT, two wrappers are developed: an XQuery/SQL wrapper (Benhlima & Chiadmi, 2003) and an XQuery/SOAP wrapper (El Marrakchi, 2009).

3. Digital libraries selection in WASSIT

Because of their increasing number and their heterogeneity, digital libraries may contain redundant information that differs by their quality characteristics. Since WASSIT answers user's queries by combining responses from different DLs, the final response quality relies on the quality of the sources involved. The perception of quality differs also from a user to another. For example, user A may ask for actual data, when user B looks for historical one.

To summarize, the concept of quality makes the difference between several DLs treating the same subject. It can be used to personalize the mediator's responses according to the user's preferences by selecting the most relevant DLs. The objective is to give a response that meets the user's quality requirements. In this section, we present our solution for DLs selection according to user's quality requirements in WASSIT. Our approach consists in building a multi-dimensional user's profile which stores the knowledge about a given user, especially his identity and quality preferences (Zaoui and al, 2009). We also construct a source profile which contains source definition, content, location, and quality characteristics (Zaoui and al, 2010). Both user's profiles and source's profiles are stored in the knowledge base. In the remaining of this section, we begin by presenting related works in section 3.1. In section 3.2, we define the quality paradigm in the domain of DLs. In section 3.3, we present our quality model to evaluate both user's quality requirements and source's quality characteristics. We use this model to select the most relevant DLs involved during the integration process in section 3.4. We illustrate our approach trough an example.

3.1 Related works

Several systems have been developed to integrate disparate and heterogeneous DLs. The majority of them addresses the problem of source selection following two approaches (Paltoglou, 2009). The first approach considers the source as a big document constructed via document concatenation, so the source selection becomes a simple problem of document retrieval. The most used source selection algorithm named CORI (Callan, 2000), is based on this assumption, GIOSS (Gravano & Garcia-Molina, 1995) and K-L divergence based algorithms (Xu & Croft, 1999) belong also to this category. The second approach considers the source as a repository of documents so the selected sources are those who are the most likely to return the maximum of relevant documents. ReDDE (Si & Callan, 2003) algorithm and the DTF (decision theoretic framework) (Nottelmann & Fuhr, 2003) give a source ranking by estimating the number of relevant documents for each query. The estimation is based on calculating a cost function which include quality and time factors. Both approaches require a source representation in their selection and ranking process. The source characteristics used are either given by the source, for example the protocol STARTS requires digital libraries to provide an accurate description of their content and quality, or discovered automatically through sampling queries (Callan, 2000). Our source selection and ranking algorithm is inspired from the second approach. We estimate the quality of each source using sampling queries and we build a quality model to perform a personalized source selection. The main contribution is that the source selection and ranking is not based on user queries but on user's profiles. The selection is performed by matching user's preferences and sources' characteristics. So, for each user, the selected set of candidate sources meets the user's quality requirements and it is also independent from the queries. These sources are used later on in the rewriting process to give a personalized response.

3.2 The quality paradigm

Many researches have been conducted to define the quality paradigm in DLs, but no single definition or standard exists. Usually, the concept of quality is the aggregation of multiple criteria organized into dimensions or categories. These dimensions may concern the quality of software, the quality of web sites, the quality of services, the quality of documents and data, and the quality of sources (Burgess and al, 2004). In the literature, there are a multitude

of quality criteria depending on the domain and the application. Taxonomy of quality indicators is presented in (Burgess and al., 2002). The authors define quality using three factors: (i) Utility, which measures the satisfaction of user's requirements; (ii) Cost, which reflects the payment given by the user and/or the system to satisfy the user's requirements; (iii) Time, which means how long the user waits to get an appropriate answer and how long the system takes to provide it. Naumann and Leser (Naumann & Leser, 1999) present other parameters concerning especially the quality of data like viability, freshness, consistency and understandability. The quality of sources is measured in most cases using factors like popularity, completeness, freshness and extent (Wang & Strong, 1997). All these quality factors could be divided in two categories. (1) Subjective quality factors, which depend on user's preferences, and vary according to the context of interaction. They are usually expressed explicitly with a score given by the user, or via a natural language using words like "good", "bad", "excellent", etc. (2) Objective quality factors, which are considered as measurable metrics, collected implicitly through statistical and data mining algorithms. To sum up, the variety of existing quality indicators makes it difficult to build an appropriate quality model. First, we need to select the most useful quality indicators that WASSIT will use to select relevant sources. Then, we have to organize them into dimensions in order to facilitate their exploitation. In the next, we give our quality model based on two dimensions. We choose the corresponding metrics and explain how to get their values.

3.3 WASSIT quality model.

To introduce our quality model, let us consider a user asking about children stories. The result may be different depending on the selected sources. If we select only a specialized source in Harry Potter editions, the result is clearly incomplete because we omit all other kid stories and novels. But if we select the most popular kids' digital library, this user may be satisfied about the completeness of the result. The result differs also depending on the user preferences. For example, user A is more interested on old stories whereas user B prefers the last published ones. From these examples, we can say that defining a quality model in a digital library integration system depends on two dimensions, which are the user's quality preferences and the source's quality characteristics.

3.3.1 User's quality preferences

We define a preference as the desired level of quality that may satisfy the user's needs. User's quality preferences are related to the quality of retrieved documents, the quality of integrated sources and finally the quality of service depending on the retrieving process and the source capabilities. In the next, we study only the user's quality preferences related to the quality of sources since our objective is to select the most appropriate ones.

We define a model where the user expresses his quality preferences in three steps. First, he chooses his desired quality criteria from a global list available in the WASSIT's user interface. Second, he gives a ranking of these criteria from the most important one to the less using weights. Weighting quality criteria helps the system to emphasize the priority of the quality criterion to satisfy. Third, he states his desired values for each criterion. Usually, user's preferences values are expressed using a numerical score in an appropriate scale, a percentage, words like "good", "bad", etc. or even a predicate (e.g., I prefer sources having recent articles than those published before 2004). In this case, the user expresses his preference about the freshness of the source. He considers that sources having only

documents published before 2004 are not fresh enough. To simplify our model, we suppose that the user states required preferences via WASSIT's user interface either by putting a score directly or by a slider on an appropriate scale. The position of the slide gives the corresponding score.

3.3.2 Source's quality characteristics

We define the source's quality characteristics as the main quality criteria that make a significant difference between data sources. In our model, the source's quality characteristics are stored in source profile. In the next, we focus on four information quality metrics which are reputation, freshness, completeness and time of response.

Reputation

Reputation, also called popularity, means the degree to which a source is in high standing (Naumann & Leser, 1999). Reputation of a source is related to several factors: (i) the quality and quantity of information and documents in the source; (ii) the authority and credibility of the source's owner (e.g., an official DL have a higher reputation than a wiki web site, a specialized DL in a given field such as computing science have a higher reputation than a DL treating all subjects); (iii) the quality of service including time of response, cost and security parameters. Indeed, a source having a good response time and a lower cost is more appreciated by the users.

Source's reputation depends on the user's judgment. It's a highly subjective criterion. For this reason, we consider that the reputation of a source S expressed by the user U is measured by a score from 1(bad reputation) to 5 (very high reputation). In the following, we denote this score by Reputation_Score(U,S).

We need now to measure the reputation of a source S. For this purpose, we define a metric called Global_Reputation_Score which is the average of all Reputation_Scores expressed by a set of users $U=\{U_1, U_2 \ldots U_n\}$. The Global Reputation Score is computed using formula 1.

$$\text{Global\_Reputation\_Score}(S) = E[\sum_{i=1}^{n} \text{Reputation\_Score}(Ui, S)/n] + 1 \quad (1)$$

Freshness

There are various definitions of source freshness in the literature, as well as different metrics to measure it. (Bouzghoub & Peralta, 2004) gives a state of the art of these definitions and presents taxonomy of metrics to measure it depending on the domain of application. For example, in data warehouse systems, one of the metrics used to measure source freshness is currency (Segev & Weiping, 1990). Currency reflects the degree of change between data extracted and returned to the user and data stored in the source. In our model, we consider that freshness refers to the age of information in the source and the update of its' content. To measure this factor, we use the Timeliness factor (Wang & Strong, 1996), which expresses how old is data in the source since its creation or update. This factor is bounded with the update frequency of the source. We define a metric called Timeliness_Score which measures the time elapsed since data was updated. For example, a "Timeliness_Score=2 years" means that the source contains documents published after 2008. We also suppose that sources give the Timeliness_Score as a meta-data in their descriptions.

Completeness

Completeness is the extent to which data is not missing and are of sufficient breadth, depth, and scope for the task at hand (Naumann and al, 1999). In other words, it expresses the

degree to which all documents relevant to a domain have been recorded in the source. Completeness of a source is also called in the literature: coverage, scope, granularity, comprehensiveness and density. For example, a scientific digital library is more complete than a non specialized one. We measure completeness using sampling queries which estimate the coverage of a source regarding some specific topic. We define a metric called Completeness_Score which represent the percentage of relevant documents returned by the source S out of the size of this source. Completeness_Score is given by formula 2, Where Size(S) is the number of documents stored in S and Size(D) is the number of documents that answer the sample queries.

$$\text{Completeness\_Score}(S) = \left(\frac{\text{Size}(D)}{\text{Size}(S)}\right) * 100 \quad (2)$$

Time of response

Time of response is the time that a source takes to answer a given query. It is calculated in seconds. Time of response could be very high if the source is saturated or doesn't have the capability to answer the query. In this case, we use our *Optimization* module to solve this problem. For the next, we suppose that the problem of source capabilities is resolved, so the time of response depends only on the communication process with the source. We use sample queries to determine this factor. Let SQ= {SQ_1, SQ_2,…,SQ_k} be the set of sample queries. For each sample query SQ_i, we measure the time of response denoted Query_Time_of_Response. The Time of Response of the source S is then computed as the maximum of all Query_Time_of_Responses using formula 3.

$$\text{Time\_of\_Response}(S) = \max_{i=1}^{k}(\text{Query\_Time\_of\_Response}(SQ_i)) \quad (3)$$

More quality factors could be found in the literature (Burgess and al, 2002). For example, understandability, credibility, precision, correctness, etc. All these factors could be added in our model easily. The user then chooses those who meet his quality requirements. In this step of work, we think that the quality factors defined are sufficient for WASSIT to make a quality aware source selection and ranking. To attempt this goal, we need to make a compromise between all defined criteria. We face two major problems. First, the source quality scores are not homogenous: we have a percentage, a time, a number. So, we need to scale the scores to make them comparable. Second, users set their quality preferences by selecting quality criteria, then stating importance weightings for each selected criterion. Finally, they state preference values for each desired criterion. So we need to select the relevant sources according to the preference values. Then, we have to rank the selected sources using the preference weightings. In the next section, we present our source selection and ranking algorithm.

3.4 Source selection and ranking algorithm

The quality of sources is measured with several criteria. Thus, source selection is a multi-attribute decision making problem (MDMP). In the literature, several methods have been developed to resolve this problem such as SAW, TOPSIS and AHP (Naumann, 1998). We choose to apply SAW (Simple Additive Weighting) (Hwang and Yoon, 1981), because it's one of the most simple but nevertheless a good decision making procedure. SAW results are also usually close to more sophisticated methods (Naumann, 1998). The basic idea of SAW is to calculate a quality score for each source using a decision matrix and a vector of preference

weights. Although SAW solves the problem of the heterogeneity of quality criteria by scaling their values, this method ranks sources considering only the user's quality preferences weights. This ranking is based on the priority and importance of quality criterion but does not consider the preference's values. Consequently, we could not select the best sources unless the user defines a limit of the acceptable source's scores or a number of desired sources. To overcome these limitations, we develop a selection and ranking algorithm that respect both the user's quality preferences weights and values. The values defined by the user correspond to the criteria thresholds. Our algorithm is performed in two stages: source selection and source ranking using SAW method. It is described in the following.

Input: $S=\{S_1,S_2,..,S_n\}$: Set of candidate sources
$Q=\{Q_1,Q_2,..,Q_m\}$: set of source's quality metrics.
$M=[v_{ij}]_{(n*m)}$: the decision matrix, where v_{ij} is the value of Q_j measured on source S_i
$W=[w_i]_m$: the vector of user's quality preference weights
$T(Q_i)$: threshold defined by user for each Q_i
Output: $S'=\{S'_1,S'_2,..,S'_k\}$: Set of selected and ranked sources
Begin
// **Stage 1:** Source Selection
1. for all Q_i select the one having the highest weight and call it Q_{max}
2. from S, select S_i having Q_{max} value ≥ $T(Q_{max})$

// **Stage 2:** Source Ranking using SAW Algorithm
1. Scale v_{ij} to make them comparable using some transformation function. With this scaling all source's quality values are in [0, 1]. We obtain a scaled decision matrix $M'=[v'_{ij}]_{(n*m)}$ where:

$$v'_{ij}= \frac{v_{ij}-\min_i(v_{ij})}{\max_i(v_i)-\min_i(v_i)}$$

2. Apply W to M'
3. Calculate sources' scores; the score of source S_i is given by: $\text{Score}(S_i) = \sum_{j=1}^{m}(v'_{ij}.w_j)$
4. Rank sources according to the sources' scores obtained in step3.

End

To illustrate our algorithm, let's consider the following example. We aim at integrating six DLs dealing with scientific field. We suppose that each DL is a single source. The integrated sources have different values of quality parameters summarized in the decision matrix (cf. Table 1).

	Global_Reputation_ Score	Timeliness_Score (years)	Completeness_Score (%)	Time_of_Response (s)
S_1	1	10	20	1
S_2	3	2	60	0.5
S_3	2	60	40	0.3
S_4	5	20	50	1
S_5	4	5	10	2
S_6	5	30	80	1

Table 1. The quality decision matrix

Consider a user who requires a Global_Reputation_Score>3. This criterion is mandatory, he also prefers sources with a Completeness_Score>30%. This criterion is desirable and he doesn't care about the other quality factors. We suppose that the user sets his preference priorities based on the following scale: {0.4: mandatory, 0.3: desirable, 0.2: not desirable, 0.1: indifferent}. The corresponding user quality preferences of this user are given in table 2.

	Global_Reputation _Score	Timeliness_Score (years)	Completeness_Score (%)	Time_of_Response (s)
Weight	0.4	0.1	0.3	0.1
Value	>3	Ø	>40%	Ø

Table 2. User's quality preferences (weights and values) (Ø means no preferred value for the criterion)

Remind that our main objective is to identify the sources that best fit with the user's quality preferences. For this purpose, we apply our source selection and ranking algorithm.

Stage 1. We select only sources having a Global_Reputation_Score>3. The remaining sources are: S_2, S_4, S_5 and S_6. Then we select only sources having a Completeness_Score>40%. The corresponding sources are: S_2, S_4 and S_6.

Stage 2. We apply SAW to the selected sources S_2, S_4 and S_6. We scale the decision matrix to make the quality values comparable. Then, we apply the vector of user's weights W to the scaled matrix. The scaled decision matrix, the vector of user's weights and the sources' scores are presented in table 3.

Sources' scores give the following ranking: S_6 is more appreciated than S_4 and finally S_2. As shown in this example, our source selection and ranking algorithm returns to the user a set of relevant sources that satisfies his quality preferences both in terms of quality weights and quality values.

	Global_ Reputation_Score	Timeliness_ Score (years)	Completeness_ Score (%)	Time_of_ Response (s)	Source Scores
S_2	0	0	0.333	0	0.0999
S_4	1	0.642	0	1	0.5642
S_6	1	1	1	1	0.9
Weight	0.4	0.1	0.3	0.1	

Table 2. Calculating sources' scores using SAW

4. Semantic interoperability in WASSIT

Semantic interoperability in DLs means the capability of different information systems to communicate information consistent with the intended meaning (Patel and al., 2005). The NSF Post Digital Libraries Futures Workshop (Larsen & Wactlar, 2003) identified it as being of primary importance in digital library research. One of the well accepted mechanisms for achieving semantic interoperability is the utilization of ontologies (Gruninger, 2002). Structure knowledge embedded in ontologies supports information retrieval and interoperability. Ontologies also help investigation of correspondences between elements of heterogeneous data sources (Shen, 2006). In WASSIT, we use ontologies to achieve semantic interoperability. Since DLs are heterogeneous, they may have local schemas or ontologies expressed in various formalism degrees, going from the informal definitions up to

rigorously formal descriptions. However, the availability of a coherent formal ontology within the mediation system facilitates semantic query rewriting by enriching terms with semantically related ones. This is a key issue for data integration. In the remaining of this section, we present in section 4.1, related works for constructing formal ontology in mediation systems. In section 4.2, we present our approach for building ontologies in WASSIT. We illustrate our approach through an example.

4.1 Related works

The most used approaches for constructing formal ontology in mediation systems are mapping and integration. We present those approaches in the following.

Ontology mapping. Mapping is a crucial process in schemas and ontologies integration as well as in semantic conflicts resolution between ontologies and between heterogeneous data sources. It is defined (Pavel & Euzenat, 2005) as being the set of operations permitting to define relationships between the elements of two schemas (or ontologies) having a semantic correspondence. We distinguish two types of mapping (Bruijn & Polleres, 2004): one-way and two-ways mapping. The first one consists in defining an expression of a destination ontology terms according to a source ontology terms, whereas the two-ways mapping operates in both directions. In our works, we are interested in the two-ways mapping, between schemas and local ontologies.

Ontology integration. The integration process consists in creating a new ontology from two or more ontologies in order to replace or to unify and then to share their vocabulary. This can be achieved using operations such as union and intersection. The intersection approach consists in producing a reduced ontology based on the terms having common semantics. The advantage of this approach is that it makes possible to obtain, easily, a reduced shared vocabulary. However, its disadvantage lies in information loss that can result from this approach. This last is used in Observer (Mena and al., 2000) where the intersection between the ontologies is measured by a percentage indicating information quantity loss during the query translation process between the different system nodes. The union approach is used when we want to get only one global ontology containing all the terms contained in these ontologies. This approach has the advantage to allow easy query rewriting since the necessary vocabulary is kept in the resulting global ontology without any information loss. It has the disadvantage to need a lot of efforts to elaborate the union task. Furthermore, adding or deleting ontologies is quite difficult. Several mediation systems use this approach. We can mention Picsel (Rousset and al., 2002) and SIMS (Arens and al., 1996).

We have adopted this last approach to build our knowledge base ontologies. But we extended it by using generalization and specialization operators. However, to palliate to its disadvantage, we propose a solution for semi-automatic integration of the local ontologies.

4.2 Building ontologies in WASSIT

To resolve semantic conflicts, we adopt hybrid architecture for the knowledge base development. Our approach combines local data sources ontologies and a global ontology that provides a shared vocabulary. This architecture offers adaptability and extensibility for new sources addition since every source has its own local ontology.

To build our ontologies, we follow the process represented in figure 2. The global ontology construction process takes place in two phases: the mapping of local schemas and ontologies, and the merging of local ontologies. The mapping of the local schemas and

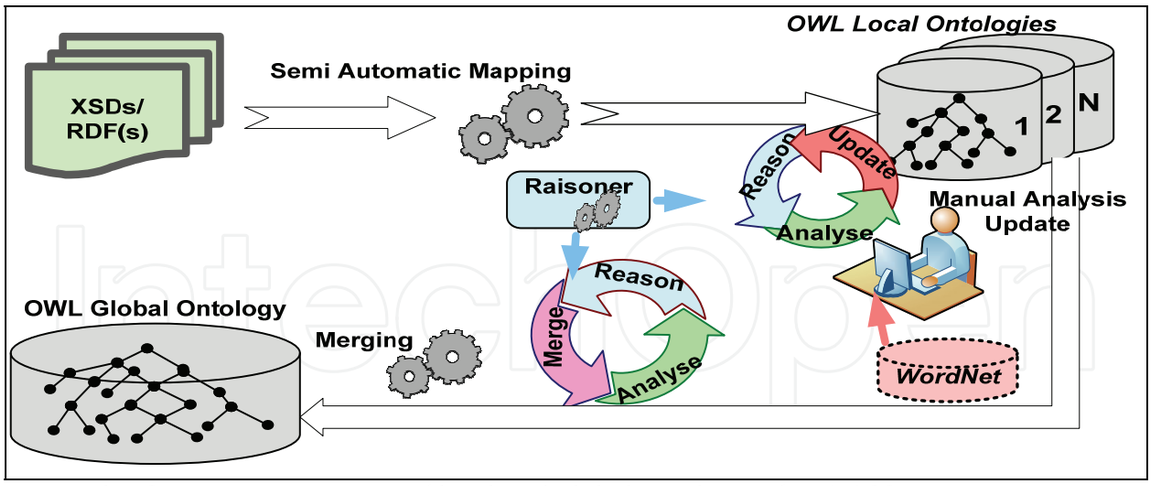


Fig. 2. The Construction process of local and global ontologies for WASSIT

ontologies of each local source in an OWL local ontology is achieved in order to permit a transparent and uniform merging of the local ontologies. At the end of this process, a mapping table is generated. It contains mapping information between local schemas and ontologies.

The merging of all OWL local ontologies resulting from the first step is achieved to build the global ontology. During the merging process, the mapping table is updated by the correspondence information between local ontologies and the global one. To illustrate our approach, we take the example of the local schemas given in figures 3 and 4. Through this example, we present the three phases of our approach, which are: *Mapping local schemas to local ontologies*, *Merging local ontologies into the global ontology* and *Consistency checking*.

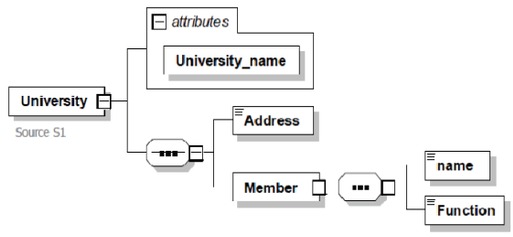


Fig. 3. Local XML Schema S_1

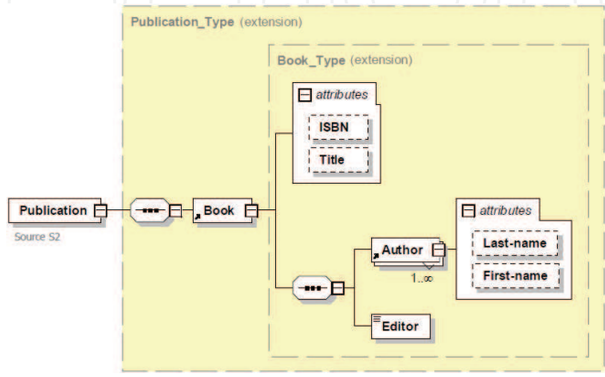


Fig. 4. Local XML Schema S_2

4.2.1 Mapping local schemas to local ontologies

Each source to be integrated is described by its local schema or its local ontology that we enrich by metadata, to add rich semantics about this data source (format, communication protocol, access rights, etc.), about its capacities and about its content.

The construction of local ontologies is accomplished by mapping local schemas and ontologies that can be represented under various formats (XML schemas, DAML-OIL, etc.). In this chapter, we limit our study to the case of the local schemas described in XML schemas.

As we adopted the OWL language for our knowledge base ontologies representation, a mapping between the XML schemas and OWL syntax is necessary (cf. Table 4). Several works for mapping between XML Schemas and OWL exist. We were inspired from the one introduced by Bohring and Auer (Bohring & Auer, 2005).

Moreover, the syntax mapping must be coupled with another one for concepts names contained in the local schemas to avoid ambiguousness that can be produced by this transformation. Indeed, an XML schema document has an ordered hierarchical structure that allows two elements to have the same identifier (name) so long as they are not in the same node. However, the order between these elements won't be taken in consideration after the mapping, because OWL doesn't define any order between properties. The OWL syntax components, rdfs:range and rdfs:domain, alone don't enable removing the generated ambiguity while transforming these elements and non-global attributes of the XML schemas toward OWL.

XSD	**OWL**
xsd:elements, containing other elements or attributes.	owl:Class, coupled with owl:ObjectProperties
xsd:elements, with neither sub-elements nor attributes	owl:DatatypeProperties
xsd:attribute	owl:DatatypeProperties
xsd:complexType	owl:Class
xsd:SimpleType	owl:DatatypeProperties
xsd:minOccurs	owl:minCardinality
xsd:maxOccurs	owl:maxCardinality
xsd:choice	combination of owl:intersectionOf, owl:unionOf and owl:complementOf
xsd:sequence, xsd:all	owl:intersectionOf

Table 4. Mapping between XML schema and OWL elements.

The definition of non ambiguous identifiers to keep a two ways mapping between the local schemas and the local ontologies is essential to permit an applicable user's query resolution. Therefore, we adopted the following process:

- The id of a local element is composed of the name of the complex type in which the element is declared +"." + its_local_name (e.g., "Book_Type.Editor").
- The id of a local attribute is composed of the name of the complex type in which the attribute is declared +".$" + its_local_name (e.g., "Book_Type.$ISBN").
- The id of a global attribute is composed of its namespace + "$"+ its_local_name.

- The id of an anonymous type is defined by the name of the element in which the definition of the type is declared +"."+ Anonymoustype (e.g., "Book_Type.author.anonymoustype").

4.2.2 Merging local ontologies into the global ontology

The goal of this phase is to generate a global ontology related to the mediated sources domain. As our objective is to achieve a virtual integration of distributed, autonomous and heterogeneous data sources, the user query must be expressed against the global ontology. Thus, this ontology must contain the whole domain concepts contained in the integrated data sources. To this end, we follow a hybrid integration of the ontologies by union completed by generalization and specialization operations. However, before performing this integration, a set of issues rises: What are the concepts and the classes to generate? What are the specializations and/or generalizations to conceive? Do these generalizations also affect properties? To answer these questions, we took into account the following constraints:

- Equivalence degree must be maintained, since a pair of concepts considered equivalent can vary a lot semantically. For example, the merging process considers "member" in the University.XSD as semantically equivalent to "Author" in Publication.xsd, although Member can be more general than author. In general, a Member cannot be an Author.
- Semantic relationship that requires one-to-many mapping (and inversely many-to-one) must be expressed correctly. For example, member.name in University.XSD is semantically equivalent to the union of the two concepts Author.first-name and Author.last-name in the Publication.XSD (Bohring & Auer, 2005).

Therefore, our approach is not reduced to a simple union of local ontologies, because we carry out specializations and/or generalizations of the concepts and properties. We use the lexical ontology WorldNet (Fellbaum, 1999) for this purpose.

4.2.3 Consistency checking

The reasoning mechanisms on ontologies allow to derive and to deduce new knowledge not described explicitly by the ontology. It can be achieved for OWL ontologies using a reasoner. The inferred information can be used to improve query resolution. In our system, we used these reasoning capacities to verify the consistency of the ontologies resulting from the mapping and merging procedures. We use the RacerPro reasoner <http://www.racer-systems.com/> to accomplish these tasks.

5. Optimizing queries over DLs with limited capabilities

In the context of digital libraries, sources may have diverse and limited query capabilities. For example, users of an online bookstore get information on books via forms. These forms allow several types of keyword based queries including search by title, subject, author, ISBN, price, etc. If we consider the web source Amazon.com <http://www.Amazon.com/>, this bookstore does not support any query that specifies conditions on the price attribute because this attribute is absent in the search form. Let us consider another web bookstore, Books.com <http://www.Books.com/>. This bookstore supports queries that specify the price attribute. However, it cannot support queries where the attribute publisher is mentioned. Consider now a third web bookstore Books-a-million.com <http://www.booksamillion.com/>. As opposed to the above mentioned online bookstores, it does not offer search neither by price nor by publisher.

As shown in the example above, DLs have diverse and limited query capabilities. These restrictions have many reasons, including the concerns of efficiency of query processing, simplicity of the query interface and security. In such situation, DLs must inform the mediator which queries they can support, so that the mediator can construct query execution plans (QEPs) that contains only feasible sub-queries. This is known as the Capability-Based Rewriting (CBR) problem. In order to be able to perform capability-based rewriting, the mediator needs formal descriptions of the query capabilities of DLs. A capability-based rewriter takes as input these descriptions and the query, and it infers query plans for retrieving the required data that are compatible with the source query capabilities. Solving the CBR typically produces more than one candidate plans for the query. Choosing the optimal plan is done using a cost model.
The problem we address in this section is how to generate efficient query plans that respect the limited and diverse capabilities of DLs in WASSIT. For this purpose, we model the source capabilities through *Capabilities Tables* and propose an algorithm to generate query plans respecting DLs capabilities. We propose also a cost model which we will use while constructing query plans. This section is structured as follows. In section 5.1, we give an overview of related works. We present in section 5.2 our solution which is made up of formalism for describing data source capabilities, a cost model and an algorithm for generating query plans.

5.1 Related works

Few mediation systems have addressed the capability-based rewriting problem. Some of these systems (e.g., GARLIC (Haas and al., 1997)) use exhaustive search methods to construct the optimal query plan according to the adopted cost model. However, the exponential complexity of these search methods limits the number of integrated data sources. Other systems, like e-XMLMedia (DANG-NGOC, 2003), verify the feasibility of the sub-queries after constructing the QEP. For each sub-query addressed to a source, the mediator checks its feasibility by consulting the source's capabilities. If a sub-query cannot be processed at a given source, the mediator attempts to download the entire source. Such an attempt is not only expensive but also may not be allowed by the source. Another category of mediation systems (e.g., DISCO (Tomasic and al., 1996)) initially ignores the limited sources' capabilities to generate possible query plans. It then checks the query plans against the sources' capabilities and rejects those containing unsupported queries. This strategy could be very expensive compared to capabilities-based rewriting as the latter ensures that the queries issued to the sources are answerable by these sources.
While developing our solution, we took into account the disadvantages that we have just quoted. Since the number of integrated DLs may be important, we use heuristic search algorithms. These algorithms construct QEPs that minimize as much as possible the cost of treatment, in a time less than that spent by the exhaustive search algorithms. In addition, the QEP generation process is based on sources capabilities descriptions. Thus, QEPs contains only feasible sub-queries. In the following, we present our solution for the capability-based rewriting problem.

5.2 Our solution

We illustrate our solution with a running example presented in section 5.2.1. Trough this example, we present our formalism for describing sources' capabilities in section 5.2.2, our

cost model in section 5.2.3 and our algorithm for constructing efficient query plans in section 5.2.4.

5.2.1 A running example

Suppose that we have three sources S_1, S_2 and S_3 and that each of them provides a local view. Let V_1, V_2 and V_3 be their local views respectively, with: V_1=(ISBN, Price, Subject), V_2=(ISBN, Author) and V_3=(ISBN, Publisher).

Sources S_1, S_2 and S_3 have limited capabilities for query processing. These capabilities are expressed as follows:

- Queries sent to S_1 must either provide the Price or the Subject field. In both cases, the set of attributes returned by the source is {ISBN, Price, Subject};
- Queries sent to S_2 must provide the ISBN field. The set of attributes returned by the source is {ISBN, Author};
- Queries sent to S_3 must provide the ISBN field. The set of attributes returned by the source is {ISBN, Publisher}.

Let BooksGV(ISBN, Price, Subject, Author, Publisher) be a global view offered by WASSIT when integrating the three data sources. BooksGV is defined as follows: $((V_1\ Join_{ISBN}\ V_2)\ Join_{ISBN}\ V_3)$. Suppose we formulate a query (Q), at WASSIT's user interface, to find all books dealing with "Linux", whose author is "Radi" and whose publisher is "Elsevier". The condition attached to the query Q is: Subject = "Linux" Λ Author = "Radi" Λ Publisher = "Elsevier".

5.2.2 Describing source capabilities

To describe source capabilities, we use a table that we call *Capabilities Table*. A *Capabilities Table* of a source S enumerates the conditions expressions that can be evaluated by S, and the set of attributes returned by S after evaluating these expressions. For example, table 5 describes capabilities of source S_1. Each row in the table describes a condition expression C that S_1 can evaluate, and the set of attributes returned by the source S_1 when processing this condition expression. For example, row 1 states that S_1 can evaluate condition expressions like (Subject= "XML") and returns the set {ISBN, Price, Subject}. *Capabilities Tables* of the integrated DLs are stored in the knowledge base of WASSIT.

	Evaluated_Attributes			**Returned_Attributes**		
Operator	ISBN	Price	Subject	ISBN	Price	Subject
Λ	0	0	1	1	1	1
Λ	0	1	0	1	1	1

Table 5. *Capabilities Table* of source S_1

We define a function called R_Attr(C) (for Returned_ Attributes) which returns the set of attributes returned by a source when evaluating a condition expression C. If a condition expression is not supported, then R_Attr(C) returns the empty set. For example, R_Attr (Price=P) = {ISBN, Price, Subject} and R_Attr (Subject=S Λ Price =P) =Ø.

5.2.3 Our cost model

In order to obtain the cost of a plan, one must have statistics about the underlying data, such as sizes of relations and sizes of domains. It is also necessary to have a cost formula to

calculate the processing cost for each implementation of each operator. Because data sources are autonomous, it may not be possible to have statistics about the sources or unreliable ones, preventing a direct application of cost models approaches developed for homogeneous systems. Several approaches have been proposed for cost based query optimization in mediation systems (DANG-NGOC, 2003). In this paper, we propose a simple cost model that we use while constructing query plans.
In mediation systems, the cost of a plan may be approximated by the sum of communication cost, source query processing costs and mediator processing cost, as expressed in formula 4.

$$cost(plan) = Communication\_cost + mediator\_cost + sources\_costs \quad (4)$$

Furthermore, in the context of web data integration, communication cost dominates source query processing costs and mediator processing cost (DANG-NGOC, 2003). If the plan consists of N sub-queries (SQ_i) executed sequentially, then the cost of a plan is expressed by formula 5. In this formula, R_i is the response corresponding to sub-query SQ_i. Minimizing the cost given in formula 5 involves reducing the number of sub-queries. This observation will be used while constructing QEPs.

$$cost(plan) = \sum_{i=1}^{N}(communication\_cost\ (SQ_i) + communication\_cost\ (R_i)) \quad (5)$$

5.2.4 Constructing query plans

When a query is formulated at the WASSIT's interface, the corresponding XQuery query is generated. This query is processed by the *Analyse & XAT Generation* module and the *Rewriting & Semantic Enrichment* module. The output of this second module is a QEP. As the global view is a join of the local sources views, the QEP generated, let be P_1, consists in sending the sub-queries SQ_1 (subject="Linux"), SQ_2 (author="Radi") and SQ_3 (Publisher="Elsevier") respectively to data sources S_1, S_2 and S_3. After retrieving results, a double join on the attribute ISBN is done at the mediator level Note that this plan is not feasible because sources S_2 and S_3 cannot answer SQ_2 and SQ_3 because of their limited query capabilities.

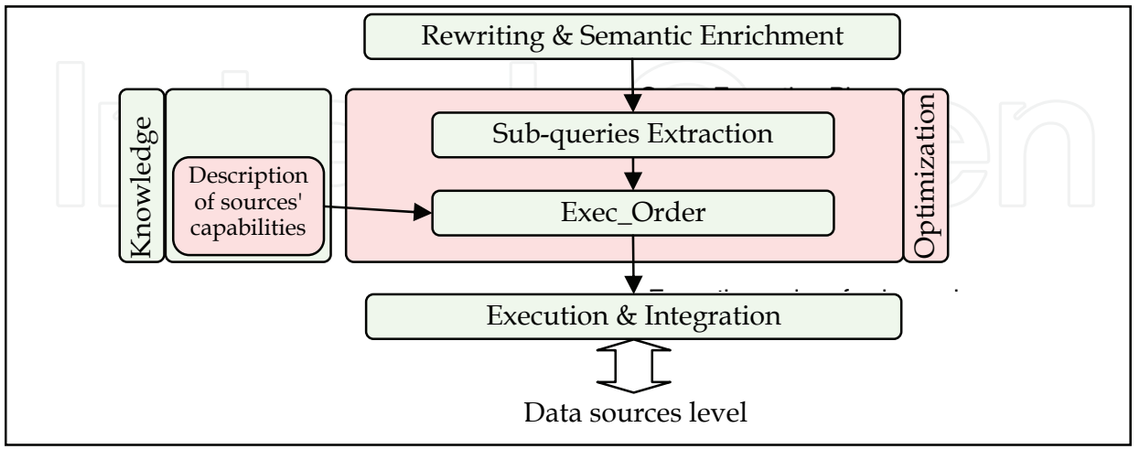


Fig. 5. *Optimization* module Architecture

Since the generated QEP contains sub-queries that are not feasible, the role of the *Optimization* module is to construct a QEP with feasible sub-queries. To this end, the first

operation performed by the *Optimization* module, when receiving a QEP, is the extraction of its sub-queries. This operation is performed by the *Sub-queries Extraction* module (cf. figure 5). The extracted sub-queries are then processed by the *Exec_Order* module. The role of this module, based on the algorithm described below, is to find an execution order of sub-queries which takes into account the limited query capabilities of the integrated data sources and that minimizes the cost of the generated QEP. To illustrate this concept of execution order of sub-queries, consider a second QEP, let be P_2. This plan consists on sending the sub-query SQ_1 (Subject = "Linux") to source S_1. For each $ISBN_i$ returned, a sub-query SQ_2 (ISBN= $ISBN_i$) is sent to source S_2. For each $ISBN_j$ returned satisfying the condition Author="Radi", a sub-query SQ_3 (ISBN= $ISBN_j$) is sent to S_3. In plan P_2, sub-queries are executed in chain. Each sub-query uses the results of the sub-query already executed.

According to our cost model, minimizing the cost of a plan involves reducing the number of its sub-queries. Suppose that source S_1 contains 30 books on "Linux" and that source S_2 contains 3 books on "Linux" whose author is "Radi". For simplicity, we count the communication cost by calculating the number of sub-queries sent to a source. For each sub-query sent to a source, we take a cost equal to 1. If we consider plan P_2, the first sub-query executed is SQ_1 (Subject="Linux"). Since each sub-query has a cost equal to 1, the communication cost of SQ_1 is equal to 1. For each $ISBN_i$ returned, the sub-query SQ_2(ISBN=$ISBN_i$) is sent to source S_2. Since S_1 contains 30 books on "Linux", 30 sub-queries are sent to S_2 with a cost equal to 30. For each $ISBN_j$ returned satisfying the condition Author="Radi", a sub-query SQ_3 (ISBN= $ISBN_j$) is sent to S_3. Since S_2 contains 3 books on "Linux" whose author is "Radi", 3 sub-queries are sent to source S_3 with a cost equal to 3. Thus, the communication cost of plan P_2 is: 1+30+3 = 34. In the cost of plan P_2, SQ_1 has the minimal cost (1) because it is executed in block. Therefore, in our algorithm we seek all sub-queries that can be executed in block. A join between these sub-queries constitute the first entity in the chain. In the next section, we present our algorithm.

Algorithm for constructing QEPs

To illustrate our algorithm, we use the running example given in section 5.2.1. Let SQ_1, SQ_2 and SQ_3 be the sub-queries extracted from the plan generated by the *Rewriting* module. Let C be the condition attached to the user query. C is: Subject = "Linux" Λ author = "Radi" Λ Publisher = "Elsevier". Let Attr(C) be the set of attributes of the condition C. This set is noted A, where A = {Subject, Author, Publisher}.

The algorithm developed is a greedy algorithm. Its idea is to find, at each iteration, a sub-query that can be executed using the attributes of set A. After executing this sub-query, the function R_Attr() (cf. section 5.2.2) is used to get the returned attributes. These attributes are added to set A. This treatment is repeated until no more sub-queries can be executed. Thus, the execution plan constructed by this algorithm is a chain of sub-queries. However, the first sub-query constituting the chain may be either a simple query or a join between multiple sub-queries. In fact, seeking the first sub-query in the chain may lead to several sub-queries. Since sub-queries at the beginning of the chain are executed in block, their execution reduces the communication cost. Therefore, these sub-queries must be executed simultaneously; a join on their results is performed at the mediator. This will constitute the first element of the chain. To sum up, the algorithm consists of three stages:

Stage 1. In this stage, the algorithm checks if all sub-queries can be executed using the attributes of set A. This test is based on *Capabilities Tables* of the integrated data sources. If so, the sub-queries are sent to data sources without any additional processing. If one of the sub-queries cannot be answered using set A, the algorithm proceeds to the second stage.

Stage 2. This stage uses the result of the first stage: all sub-queries that can be answered using the attributes of set A, form the first element of the plan. Thus, these sub-queries will be executed in parallel. A join of their results is performed at the mediator level. The attributes returned after the execution of these sub-queries are added to set A.
In the running example, only the sub-query SQ_1 can be executed. It is the first sub-query in the chain. The attributes returned by SQ_1, which are ISBN and Publisher are added to set A. A becomes = {Subject, Author, Publisher, Price, ISBN}.
Stage 3. In this stage, we seek among the remaining sub-queries, a sub-query that can be executed using set A. If this sub-query exists, the set A is enriched with the attributes returned after its execution. In the example, SQ_2 is selected and A becomes A = {Subject, Author, Publisher, Price, ISBN}. The same process is repeated until no more sub-queries must be executed. If at a given step, no sub-query can be executed using the attributes of set A, then there is no plan to execute the target query. Below, we give a formal description of the algorithm.

```
Input:   Set of sub-queries SQ={SQ1, SQ2,.., SQn}
         Set A
Output:  Plan (if exists)
Begin
  // Stage 1
1.   Plan ← Ø, B ← Ø
2.   For each (SQi )i=1 to n
3.        if SQi can be answered using A
4.             B ← SQi
5.        if (B==SQ)
6.             Plan ← {SQ1, SQ2,.., SQn}
7.        Else
8.             Plan ← Ø
  // Stage 2
9.   If Plan == Ø
10.       Plan ← Plan . [B]    // B contains sub-queries that will be joined
11.       A ← A U {attributes(B)}
12.       SQ ← SQ – {B}
  // Stage 3
13.  While SQ ≠ Ø
14.       For each SQi
15.            if SQi can be answered using A
16.                 N←SQi
17.                 Break
18            Else
19.                 Return (Ø)     //The plan does not exist
20.       Plan ← Plan . [N]
21.       SQ ← SQ – {N}
22.       A ← A U {attributes(N)}
23.  Return (Plan)
END
```

6. Performance evaluation

In this section, we analyze the performances of our framework WASSIT, when integrating DLs, through simulation. The performance index that we evaluate is the response time. For this purpose, we present the key parameters that have an impact on the response time in section 6.1. Then, we present and discuss the most important results in section 6.2.

6.1 Performance parameters

We study the influence of the key parameters on the response time, which is defined as the time elapsed between submitting a query to WASSIT and getting a response. In mediation systems, response time may be important (Travers, 2006) (Langegger and *al*., 2008). This is due to communication cost, source query processing costs and mediator processing cost. Furthermore, additional processing time is introduced by our *Optimization* module. But this is still tolerable since we give to the user the guaranty to get a response in a finite time. Without our *Optimization* module, the mediator may not answer some queries because of the limited query capabilities of the underlying DLs. Response time depends also on the bandwidth of the communication network, between mediator and data sources. The system load, which lies principally on queries frequency and size of sources' responses, has also an impact on the response time. We summarize these parameters in table 6.

Parameter	Meaning
N	Number of integrated sources
R_Size	Response size (Bytes)
F = 1/T	Queries frequency: number of queries addressed to WASSIT per time unit. T is the period of query arriving (seconds)
BW	Network Bandwidth

Table 6. The key parameters for studying the system performances

6.2 Results analysis

We remind that our objective is to evaluate the performances of WASSIT. Due to the lack of space, we present only the most important simulation results. In the first simulation scenario, we study the impact of the *Optimization* module on the response time. (cf. figure 6). In the second scenario, we evaluate the response time for different values of bandwidths (cf. figure 7). The third scenario consists on analyzing the influence of the responses' sizes variation and the period of query arriving on WASSIT's performances (cf. figure 8 and figure 9). We present results analysis below.

Figure 6 shows an exponential increase of response time depending on the number of integrated sources. This is due to the processing time introduced by the *Optimization* module. We also deduce that the system has good performances if we integrate less than six sources. As the number of sources increases, these performances degrade and for more than 20 sources, the system begins to crush.

Figure 7 shows that response time decreases when the bandwidth increases. Indeed, increasing the bandwidth reduces communication cost, which is the most penalizing cost in mediation systems. Consequently, communication networks with good bandwidth would help to integrate more data sources.

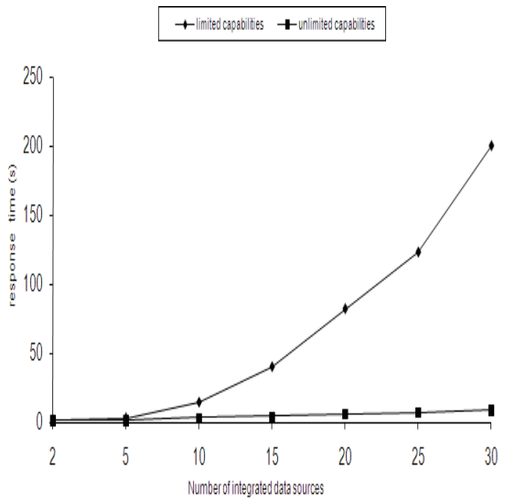


Fig. 6. Response time depending on N

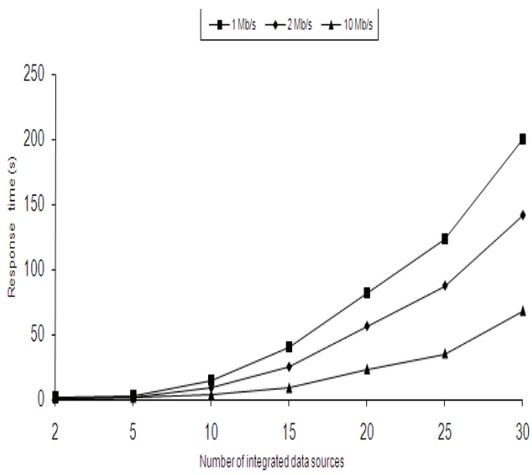


Fig. 7. Response time for 3 values of BW

Figure 8 shows that reducing responses sizes implies reduction of response time. Indeed, the reduction of responses sizes minimizes communication cost. Thus, when the responses returned by the integrated data sources have small sizes, the system performances are improved.

Figure 9 shows that, for periods greater than 12 seconds, the response time remains constant. This indicates that for these periods, the system goes idle waiting for new queries. The figure shows also that for periods less than 4 seconds, the response time increases exponentially. This induces performance deteriorating and the mediator becomes a bottleneck for the system.

To summarize, to improve WASSIT's performances, we can either reduce the number of integrated sources or reduce the system load. This can be performed if responses sizes are smaller, and if queries frequency is reduced. In addition, communication networks with good bandwidth would help to have better performances.

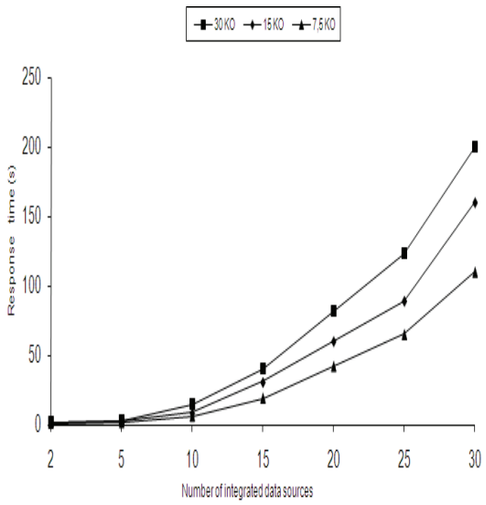


Fig. 8. Response time for 3 values of R_Size

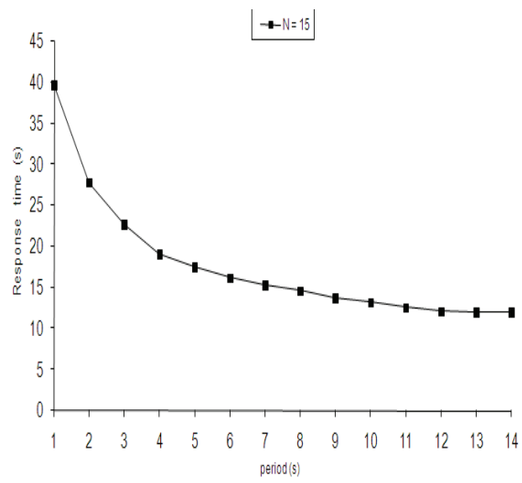


Fig. 9. Response time depending on T

8. Conclusion and future works

In this chapter, we use the mediation framework WASSIT to integrate disparate DLs. The main challenges faced in this integration are selecting DLs according to the user quality requirements, dealing with semantic interoperability and constructing query plans with respect to the limited query capabilities of the underlying DLs. To select DLs, we define a quality model based on two dimensions, which are the user's preferences and the source's quality parameters. We develop an algorithm that selects and ranks the sources respecting the user's preferences. The ranking is performed using the well known SAW method. To deal with semantic interoperability, we use ontologies. To build our ontologies, we apply the union approach to local ontologies. We improve the union approach by carrying out specializations and/or generalizations of the concepts and properties. For constructing

query plans respecting the limited query capabilities of DLs, we develop a formalism for describing sources capabilities, a cost model and an algorithm for constructing query plans.
We perform simulations to evaluate our system performances. The results show an acceptable response time for a given number of integrated sources. However, in some situations, the system becomes a bottleneck when the number of integrated DLs is important. This may occurs in the context of very large digital libraries. Despite the fact that WASSIT's reduces the number of integrated sources via sources selection, the number of selected sources may still high inducing performance degradation. To deal with this limitation, we plan to extend our work in two directions:

- **Using a hierarchy of mediators.** In this architecture, an instance of WASSIT integrates other instances of the same mediator. Each integrated mediator will integrate DLs. We believe that this solution will improve performances because it allows processing parallelization.
- **Using a peer-to-peer architecture.** In this architecture, a peer's network is formed. Each node in the network contains an instance of WASSIT integrating DLs. The absence of a central node avoids bottlenecks. We anticipate that this solution will give us good performance.

9. References

Arens, Y.; Knoblock Craig A. & Hsu, C. (1996). Query Processing in the SIMS Information Mediator, *Proceedings of Advanced Planning Technology*, *AAAI Press*, California, USA.

Benhlima, L. & Chiadmi, D. (2003). XQuery–SQL wrapper for integrating relational data sources", *Proceedings of COPSTIC'03*, pp 54–57, Rabat, Morocco, Dec. 11–13.

Bohring, H. & Aue, S. (2005). "Mapping XML to OWL Ontologies", *Proceedings of 13. Leipziger Informatik-Tage (LIT 2005), Lecture Notes in Informatics (LNI),* Vol. 72, pp. 147-156, ISBN 3-88579-401-2.

Bouzeghoub, M. & Peralta, V. (2004). A Framework for Analysis of Data Freshness, *International Workshop on Information Quality in Information Systems (IQIQ'2004), co-located with SIGMOD Conference*, Paris, France.

Bruijn, J. & Polleres, A. (2004). Towards an Ontology Mapping Specification Language for the Semantic Web, *Digital enterprise research institute deri,* Technical Report 2004-06-30.

Burgess, M.; Alex Gray, W. & Fiddian, N. (2002). Establishing Taxonomy of Quality for Use in Information Filtering, *Actes de la 19th British National Conference on Databases (BNCOD)*, pp. 103-113, Sheffield, UK.

Burgess, M-S-E; Gray, W.A. & Fiddian, N-J. (2004). Quality Measures and the information consumer. *Proceedings of the Ninth International Conference on Information Quality (ICIQ-04)*.

DANG NGOC, T-T. (2003). *Integration of semi-structured data with XML*, Ph.D. dissertation, Versailles Saint-Quentin-en-Yvelines Uniersity, France.

El Marrakchi, M. (2009). *Mise en place d'un adaptateur XQuery/SOAP pour l'interrogation des Web services à partir d'un système de médiation*. M.S Thesis, Computer Sciences Department, Mohammadia Engineering School, Rabat, Morocco.

Fellbaum, C. (1999). *WordNet: An electronic lexical database.* Cambridge, Massashusset, MIT Press.

Fernández, M.; Malhotra, A.; Marsh, J., Nagy, M. & Norman, W. (2007). XQuery 1.0 and XPath 2.0 Data Model (XDM) W3C Recommendation 23, January 2007, Available from http://www.w3.org/TR/xpath-datamodel/

Garcia-Molina, H.; Papakonstantinou, Y.; Quass, D.; Rajaraman, A.; Sagiv, Y.; Ullman, J-D.; Vassalos,V. & Widom, J. (1997). The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, Vol. 8, No. 2, pp. 117-132.

Gounbarek, L.; Benhlima, L. & Chiadmi, D. (2009). Data Integration System: towards a prototype, *The seventh ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-09)*, Rabat, Morocco.

Gravano, L.; Chang, C.; Garcia-Molina, H. & Paepcke, A .(1997). STARTS: Stanford proposal for internet meta-searching, *Proceedings of the ACM-SIGMOD International Conference on Management of Data*.

Gruninger, M. & Lee, J. (2002). SPECIAL ISSUE: Ontology applications and design. *Communications of the ACM*, Vol. 45, Issue.2, pp. 39–41.

Haas, L.; Kossmann, D.; Wimmers E.; & Yang J. (1997). Optimizing queries across diverse data sources, *Proceedings of 23rd International Conf. on Very Large Data Bases, VLDB'97*, Athens, Greece, pp. 276–285.

Harrati, R. & Calabretto, S. (2006). Un modèle de qualité de l'information. *Actes des journées Extraction et Gestion de Connaissances (EGC),* p.299-304, Lille, France.

Hasselbring, W. (2000). Information System Integration: Introduction. *Communications of the ACM,* Vol. 43. Issue. 6, pp. 32-38.

Hwang, C.L. & Yoon, K. (1981). Multiple Attribute Decision Making: Methods and Applications. Springer-Verlag, Berlin, Heidelberg, New York.

Langegger, A.; Woß, W. and Blochl, M. (2008). A Semantic Web Middleware for Virtual Data Integration on the Web, *In proceedings of the 5th European Semantic Web Conference*, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5.

Larsen, R-L. & Wactlar, H.D. (June 2003). Knowledge Lost in Information. *Report of the NSF Workshop on Research Directions for Digital Libraries*, (June 15-17), Chatham, MA, National Science Foundation Award No. IIS-0331314. http://www.sis.pitt.edu/~dlwkshop/

Lenzerini, M. (2002). Data Integration: A Theoretical Perspective. *Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2002)*, pp. 233–246, Madison, Wisconsin, US.

Mcguinness, D-L. & Harmelen, F-V. (2004). OWL Web Ontology Language Overview. W3C Recommendation 10 February 2004, Available from http://www.w3.org/TR/2004/REC-owlfeatures-20040210/

Mena, E.; Illarramendi, A.; Kashyap, V. & Sheth, A-P. (2000). OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-Existing Ontologies. *Journal of Distributed and Parallel Databases*, Vol. 8, No. 2, pp. 223-271.

Moujane, A. (2006). *La sémantique fondée sur les ontologies pour la plate-forme WASSIT. M.S. thesis,* Computer Sciences Department, Mohammadia Engineering School, Rabat, Morocco.

Naumann, F.; Leser, U. & Freytag, J.C. (1999). Quality –driven integration of heterogenous information systems. *Proceedings of the 25th International Conference on Very large Data Bases (VLDB'99),* pp. 447-458.

Nottelmann, H. & Fuhr, N. (2003). Evaluation different methods of estimating retrieval quality for resource selection. *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Paltoglou, G. (2009). *Algorithms and strategies for source selection and results merging (Collection fusion algorithms) in distributed information retrieval systems.* PhD thesis, Department of Applied Informatics, University of Macedonia.

Patel, M.; Koch, T.; Doerr, M.; & Tsinaraki, C. (2005). Semantic Interoperability in Digital Library Systems, *report of DELOS2 Network of Excellence in Digital Libraries*, Deliverable D 5.3.1.

Pipino, L.; Lee, Y. & Wang, R. (2002). Data quality assessment. *Communications of the ACM,* Vol. 45, No. 4, pp. 211-218.

Rousset, M-C.; Bidault, A.; Froidevaux, C.; Gagliardi, H.; Goasdoué, F.; Chantal, R. & Safar, B. (2002). Construction de médiateurs pour intégrer des sources d'information multiples et hétérogènes : le projet PICSEL. *Revue I3 : Information - Interaction – Intelligence,* Vol. 2, No. 1, pp. 5-59.

Rundensteiner, E.; Koeller, A. & Zhang, X. (2000). Maintaining Data Warehouses over Changing Information Sources, *Communications of the ACM*, Vol. 43, No. 6, pp. 57-62.

Segev, A. & Weiping, F. (1990). Currency-Based Updates to Distributed Materialized Views, *Proceedings of the 6th International Conference on Data Engineering (ICDE'90),* Los Angeles, USA.

Shen, R. (2006). *Applying the 5S Framework To Integrating Digital Libraries,* PhD dissertation, Virginia Polytechnic Institute and State University, Virginia, US.

Shvaiko P. & Euzenat, J. (2005). A survey of schema-based matching approaches. *In Journal on Data Semantics IV*, pp. 146-171.

Si, L. & Callan, J. (2003). Relevant document distribution estimation method for resource selection, *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Strong, D.; Lee, Y. & Wang, R. (1997). Data quality in context, *Communications of the ACM,* Vol.40, No 5, pp. 103-110.

Tomasic A.; Raschid L., & Valduriez P. (1996). Scaling heterogeneous databases and the design of DISCO, *Proceedings of the 16th International Conf. on Distributing Computing Systems (ICDCS),* pp. 449–457, Hong Kong.

Travers, N. (2006). *Optimisation Extensible dans un Médiateur de Données Semi-Structurées.* Ph.D. dissertation, Université de Versailles Saint-Quentin-en-Yvelines, France

Wadjinny, F. & Chiadmi, D. (2006). XML Algebra for SIRENE, *Proceedings of MCSEAI'06*, pp 63-568, December 7-9, Agadir, Morocco.

Wang, R. & Strong, D. (1996). Beyond accuracy: what data quality means to data consumers. *Journal on Management of Information Systems*, Vol. 12, No. 4, pp. 5-34.

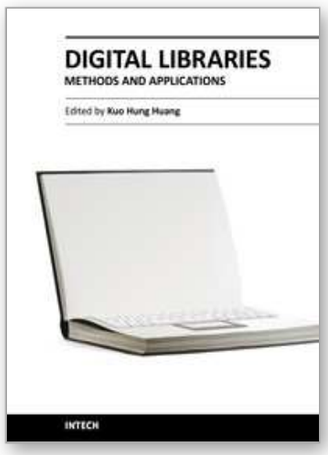
Wiederhold, G. (1992). Mediators in the Architecture of Future Information Systems. *In IEEE Computer Journal*. Vol. 5, No. 3, pp 38-49.

Xu, J. & Croft, W.B. (1999). Cluster-based language models for distributed retrieval. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zaoui, I.; Wadjinny, F.; Chiadmi, D. & Benhlima, L. (2009). Construction d'un profil utilisateur pour un médiateur de bibliothèques électroniques, *Proceeding of WOTIC*, Agadir, Morocco.

Zaoui, I.; Wadjinny, F.; Chiadmi, D. & Benhlima, L. (2010). Towards a personalized data source selection in the context of mediation systems, *Proceeding of the third International Conference on Web and Information Technologies,* Marrakech, Morocco.

Zellou, A. (2008). *Contribution to the LAV rewriting in the context of WASSIT, a resources integration framework.* Ph.D. dissertation, Computer Sciences Department, Mohammadia Engineering School, Rabat, Morocco.



Digital Libraries - Methods and Applications

Edited by Dr. Kuo Hung Huang

ISBN 978-953-307-203-6
Hard cover, 220 pages
Publisher InTech
Published online 04, April, 2011
Published in print edition April, 2011

Digital library is commonly seen as a type of information retrieval system which stores and accesses digital content remotely via computer networks. However, the vision of digital libraries is not limited to technology or management, but user experience. This book is an attempt to share the practical experiences of solutions to the operation of digital libraries. To indicate interdisciplinary routes towards successful applications, the chapters in this book explore the implication of digital libraries from the perspectives of design, operation, and promotion. Without common agreement on a broadly accepted model of digital libraries, authors from diverse fields seek to develop theories and empirical investigations that to advance our understanding of digital libraries.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen