Designing and Querying Web Services Communities

Jalel Akaichi, Hela Limam, Mourad Khayati

LI3 laboratory, ISG University of Tunis, 41 Avenue de la Liberté Bouchoucha, Bardo, TUNISIA

Summary
The incredible growth of the information space, the hard competition between enterprises populating it, the need to collaborate and to share knowledge guiding to better decisions are factors that push Web Services providers to join each other into Web Services Communities. In fact, solutions proposed to tackle communities' management lack of a formal and clear design which may alter the system evolving and maintenance. For these reasons, we propose to design and to implement a system able to manage Web Services Communities. Moreover, these communities are built with the goal to be queried transparently and easily by users which aim to satisfy their informational needs in a satisfactory time and in a pertinent retrieval .For an efficient query processing we developed an advanced caching mechanism to reduce the query response time.

Key words:
Web Services Communities, Mediator, Management, Queries Optimization.

1. Introduction

With the advance of Web Services technologies and the mergence of Web Services into the information space, tremendous opportunities for empowering users and organizations appear in various application domains including electronic commerce, travel, intelligence information gathering and analysis, health care, digital government, etc.
However, the technology to organize, search, integrate these Web Services has not kept pace with the rapid growth of the available information space. The number of Web Services to be integrated may be large and continuously changing.
The ubiquitous need for Web Services integration across heterogeneous information sources pushed Web Services providers to join each others into Web Services Communities.
In fact, Web Services Communities introduce two main problems, a design problem resulting from the lack of a generic tool for community management and the absence of interoperability among different communities support platforms, and a querying problem since we are facing more and more exigent users.

A critical challenge therefore is to design a system able to manage communities taking into account many tasks such as creating and updating communities, building relationships between them and to offer an efficient query processing mechanism.
Our solution is to design and to implement a mediator able to manage Web Services Communities while addressing all of these issues.
Moreover, in order to optimize the processing of user queries submitted to communities through our mediator, we propose a mechanism for queries caching. This will decrease the whole cost by identifying easily communities' members able to provide answers for users' queries. We also address the issue of maintaining the cache to remain up to date according to changes that may affect communities' structures.
This paper is organised as follows. In section 2, we first present the related work. Section 3 describes functionalities and architectural issues of our system. Section 4 details our management approach. Next we introduce the query processing in section 5. Section 6 presents the query optimization and section 7 illustrates the cache maintenance. We describe our system implementation in section 8. Finally, section 9 presents our conclusions and future work.

2. Related Works

Two major areas of related research are building and querying Web Service Communities.
A community therefore, is an aggregator of service offered with a unified interface. It is intended as a means to support the composition of a potentially large number of dynamic Web Services [1].
In the specific context of catalog integration, existing e-catalogs and data integration approaches typically rely on a global schema integration style [2, 3, 4 and 5].In fact, two main approaches have been investigated depending on how the relationship between the global schema and the sources is defined: the GAV (Global As View) approach [6] defines the global schema as a set of views over the data sources while the LAV (Local As View) approach [7,8] defines the sources as a set of views over the global schema. Each of these two approaches has advantages and drawbacks. It is well known that query rewriting is easier in the context of a GAV approach while a LAV approach

Manuscript received October 5, 2007
Manuscript revised October 20, 2007

is more flexible (i.e., it enables easy addition of new sources).

WS-CatalogNet [9] follows a LAV approach to define the mappings between community ontology and the exported descriptions of the community members (i.e., member definitions are expressed as views over the community ontology). As mentioned before, a GAV approach, where the development of an integrated schema requires the understanding of both structure and semantics of all schemas of sources to be integrated, is hardly scalable because of the potentially large number and dynamic nature of available e-catalogs.

WS-CatalogNet [10, 11] consists of a set of integrated tools that allow for creating communities, registering catalog members, creating peer relationships between communities, querying individual communities and routing queries among communities. It is a hybrid of peer-to-peer and Web Services technologies. Given the highly dynamic and distributed nature of e-Catalogs, a novel approach that involves techniques such as peer-to-peer and Web services will become increasingly attractive in building e-catalog portals.

The general architecture of WS-CatalogNet[4] contains three main components: Community Manager, Member Manager and Cooperative Query Manager. The Community Manager is used to create communities and build peer relationships between communities. The Member Manager supports registering individual e-catalogs into communities. The e-catalogs are either already Web Services, or converted to Web Services through the Member Manager. The cooperative Query Manager processes the query which requires locating e-catalogs capable of answering the query. The latter component allows us to introduce the second research field: Querying Web Services communities which will be detailed in the following.

In fact, Query routing among peers is discussed in [12]. This system supports limited querying capabilities (e.g., keyword based search).However, effective data sharing requires support for more structured querying capabilities to exploit the inherent semantics in data [13, 14]. Authors in [15] Propose a super-peer based routing in RDF-based P2P information sources. The proposed approach focuses on indexing RDF resources.

Concerning online catalog products, most portal sites provide customers with two ways of searching [11]: browsing via categories, searching using keywords. The two approaches can also be combined by restricting keyword searches to a category.

In general, the portal will return a product as a “match”, if any part of the description of the product contains the keyword. Answering queries based on keywords has obvious Short-comings [16]. The query model in current web portals (e.g., Amazon.com) does not allow users to query precisely using rich descriptions of the products. For example, users can only ask for “products related to laptop”, rather than “a laptop with SONY as a manufacturer, more than 2 year warranty”.

Piazza [17] considers the issue of schema mediation in P2P environments. It uses a relational model to describe peer schemas. It also proposes a query reformulation algorithm for the proposed mediation framework. Authors in [14] propose the notions of Mutant Query Plan (MQP) and multi-hierarchy namespaces to support query processing in P2P environments. A MQP includes verbatim XML encoded data, references to actual resource locations (URL) and references to abstract resource names (URN). A peer can mutate an incoming MQP by either resolving URNs to URLs, or substituting a sub-plan with the evaluated XML encoded data. The framework proposed in [17] allows for unifying Web services and peer-to-peer computing technologies together. The framework is based on the idea of super-peers which act as registries of web services that have similar behaviors. The super-peers could be compared with the concept of communities in WS-CatalogNet. However, different types of relationships between super-peers and specification of forwarding policies are not discussed in [17].

In WSCatalogNet [18], each peer can be viewed as a domain specific data integration mediator which holds metadata and registry information about its members. Queries are processed by the members of a peer, but routing of the queries is a responsibility of the peers. It should be noted that the purpose of the query routing is to identify a set of members that, when put together, can satisfy all constraints required by a query. Hence, a routing takes place before the actual query process. Once a set of members (not necessarily from the same peer) are identified, queries are sent to each member in the set for processing. The results are combined by the original community.

Since a community does not store product data locally, processing the query requires locating catalogs that are capable of answering the query. The authors in [9] propose a cooperative query processing technique that consists of two steps:

- Identify best combinations of members whose query capabilities, when put together, satisfy the constraints expressed in the query.
- Resolve the query by sending it to the selected combination of members.

A query rewriting algorithm is developed and adopted by the authors [19]; Best Query Rewriting (BQR) [20]. This algorithm identifies which part of the query can be answered by local members of the community and which part of the query cannot (hence, needs help of peers).

3. The System Architecture

We describe in these part functionalities and architectural issues of the Web Services Communities management system (see fig 1). Our system can be seen as a mediator used to process users' queries and identify relevant share knowledge. When the query is received, the mediator analyzes the query and locates relevant sources, and presents the answer that is merged, assembled or redefined. Furthermore, it may keep, fully or in part, the result of frequent queries.

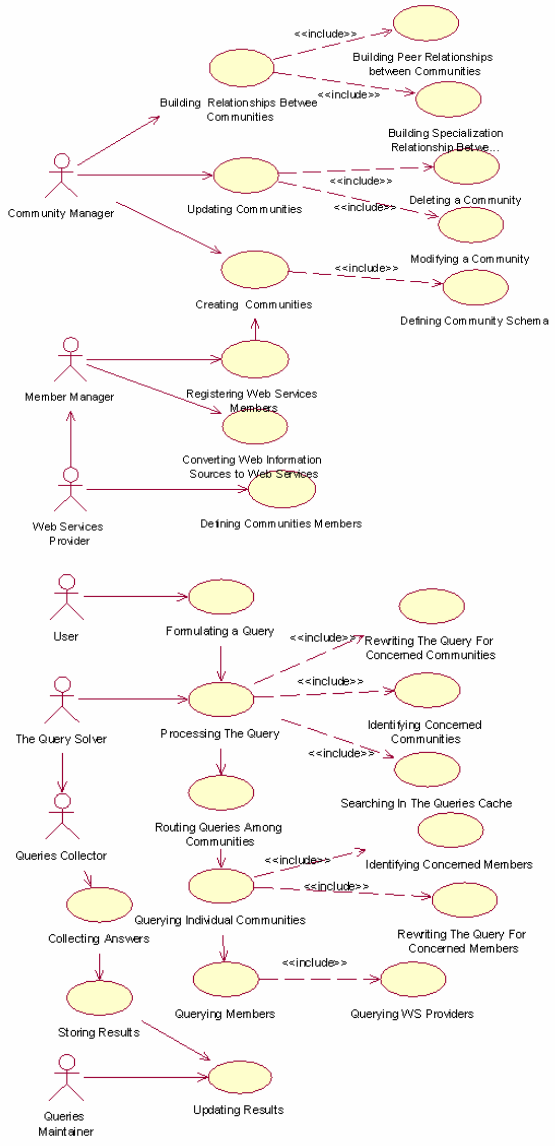

Fig. 1 The system functionalities.

The mediator is in charge of community creation, community update, community members' participation and the research of the best provider for a requested service. The mediator architecture includes five main components (see fig 2). The Community Knowledge Base (CKB) responsible of the communities' management, The User Queries Knowledge Base (UQKB) where user queries are stored and a Query Solver (QS) which main role is to extract and, route sub queries destined to specific communities. The Semantic cache (SC) saves the queries results in a queries cache for an immediate and future use of these queries. And the cache maintainer (CM) is responsible for the coherence and the availability of the cache.

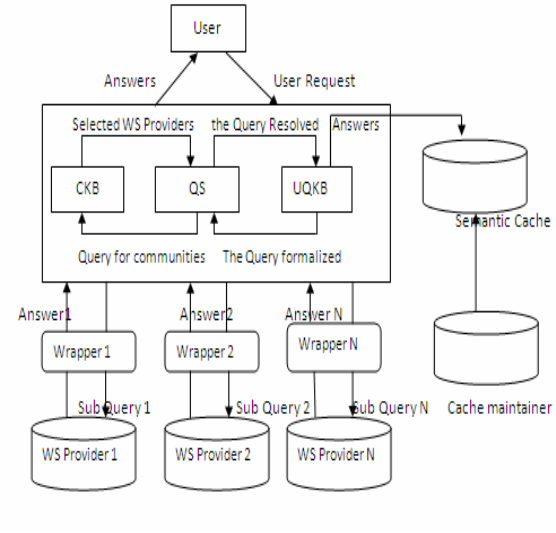

Fig. 2 The system architecture.

3.1 The Communities Management components

The communities' management component constitutes an intermediary between Web Services providers and the mediator. Web Services providers can join the mediator via Wrappers which convert them to Web Services, then Web Services are integrated into communities. Communities are managed in the mediator thanks to the Community Knowledge Base.

The community Knowledge Base (CKB): The Community Knowledge Base is used in order to provide context for queries. It represents all the entities that exist on the system, these entities become source of knowledge for the users of the mediator .The Community Knowledge Base support the community creation, the community management and building relationships between communities. It contains descriptions of communities using the structure (CId, CName, WSCList), where:

- Cid describes the community identifier.
- CName represents the community name.

- WSCList symbolizes the Web Services list from which communities are created.

Moreover we include into the CKB peer constraints and specialization relationships between communities.

SubCommunityOf relationship: It represents specialization between two communities' domains (for example, Hospital is a subcommunity of Medical Institutions). We assume each community has at least one supercommunity but can have no subcommunity.

PeerCommunityOf: When a user cannot find (or is not satisfied with) information from a community, she or he can refer to other communities considered as its peers. (For example, Patient is a peer community of Examination). Communities can also forward queries to each other via a PeerCommunityOf relationship. In fact, Peer relationship is based on the similarity between categories or attributes of different communities.

3.2 The Query Processing Components

The goal of our mediator is to satisfy user's queries. The achievement of this goal while using an optimized manner requires the following components:

The User Queries Knowledge Base (UQKB): The User Queries Knowledge Base offers a space to a user in which he is able to retrieve information stored in the system, communicate or interact. The User Queries Knowledge Base performs the task of the description and the formalization of the query.

The Query Solver (QS): The main functionality assumed by the query solver consists on processing the query ad answering it which requires locating Web Services capable of giving an answer to the query by routing the query among communities then querying individual community .The result of this step is finding the combination of members satisfying the query constraints

The Semantic Cache (SC): The cache is indirectly accessed, through the query. Facts and concepts are stored in a format that treats relations and concepts as first-order objects and that is therefore very flexible with regard to changes and amendments of the information sources.

The Cache Maintainer (CM) The cache maintainer is in charge of managing the semantic cache space when changes occur in communities by replacing them by their peers.4. Web Services communities Management

4.1 Communities Creation

A community is a container of Web Services of a specific domain that cater for similar customers needs (e.g. Health Care Community). A community is associated to community schema that provides description of a specific domain of interests. The community schema is described in terms of categories for the domain it represents. A category is described by a set of attributes. Categories within community schema may be inter-related via the specialization relationship.

4.1.1 System initialization

The communities' creation process starts when a community Administrator send a request for creating a community and specifies its name. Hence, the Community Manager prepares a community structure of the schema inside which Web Services will be integrated progressively. For example, to create a Health Care Community, the Community Manager specifies that sub communities are Examination, Medical institutions, Patients, Pathologies and Doctors. Thereafter, the Community Manager feeds the empty structure with Web Services by locating related We Services and specifying its category and for each category a set of attributes.

4.1.2 Communities' Members' Registration

The descriptions of community members are expressed in terms of the community schema; thereby our system follows a LAV [8, 9] approach for integrating a community schema with the descriptions of its members as shown in figure 4.4. This approach is especially useful as a given community can have a potentially large number of Web Services. Clearly, a GAV integration approach [6], where the development of a community schema requires the understanding of both structure and semantics of all descriptions of community members, is hardly applicable in Web Services environments because of the dynamic nature and size of the Web. Furthermore, the addition of a new member to the community requires only the provision of the exported description of the member. It does not require any changes to the community schema.

In fact, the member subscription takes in general two forms depending on the Web Provider: If Web Information sources are Web Services they directly apply for subscription in communities. They indicate through the Member Manager which category in the community schema they belong to, then for each category chosen they specify the attributes they support. Else they are converted to web services in order to be accessible through a community.

4.2 Communities Update

As Web Services Communities evolve in a dynamic environment, changes can occur and affect as well as their schemas as their contents. For theses reasons an update process in to be applied permanently. In fact, the update takes in general two main forms: Deletion or modification.

4.2.1 Communities deletion

The Community Manager deletes the community which does not contain any Web Service. As it can't satisfy requests, users constantly leave it without performing any further action. When the community administrator asks for community deletion, the Community Manager counts the Web Services in the community. If the latter does not contain any Web Service, the community manager has to delete it. Deleting a community is removing its category and its associated attributes. The Deletion is logic.

4.2.2 Communities modification

Community modification consists on adding or updating members or adding new Web Services manually or automatically.

5. Querying Web Services Communities

Processing the query requires locating communities' members that are capable of answering the query. These members could be local members of the community or the members of the peers.
The cooperative query processing technique consists of two steps:

- Identify combination of members whose query capabilities, when put together, satisfy as much as possible the constraints expressed in the query,
- Answer the query by sending it to the selected combination of members.

The first step uses a query rewriting algorithm, called Best Quality Rewriting (BQR), which allows identifying: (i) the part of the query can be answered by local members of the community and (ii) the part of the query that cannot by be answered by local members and hence can be forwarded to peer communities. The algorithm takes as input the community definition, member definitions and the query (all in their class descriptions format) and produces the following outputs:
Qlocal: the part of the query Q that can be answered by the community's local members. It also gives the combination of the local members that can answer all (or part of) the query. For each selected local member, we compute the part of the query to be sent to the member.
Qrest: the part of the query that cannot be answered by the local members. This part of the query will be forwarded to peers. It should be noted that the expected answers of the forwarding is the combination of the external members that are capable of answering the part of the query.

6. Queries Optimization

As shown above, our system accesses distributed Web Services Communities in order to satisfy a user query. However an efficient query processing requires an advanced caching mechanism to reduce the query response time.

6.1 The Semantic Caching Approach

The idea of semantic caching [21] has been proposed recently. The approach uses semantic information to describe cached data items with the goal of exploiting semantic locality to improve query response time. That is, semantic caching suggests to cache users' queries and correspondingly to organise the answers (instead of pages or tuples) by their query descriptions.
Based on these query descriptions, a semantic caching system determines whether a given input query Q is logically contained or overlapping with a cached query V.

When a query is posed at a community with a semantic cache, the query is split into two pieces: (1) a probe query, which retrieves the portion of the answer available in the local cache, and (2) remainder query, which retrieves the missing portion of answer from communities. If the remainder query is not null (i.e., the query asks for answer that is not cached), the remainder query is sent to the others communities and processed there.

6.2 The Query Processing using the Cache

The query processing using the semantic cache follows the next steps:

- The User uses a community to express queries; he submits the query to the current community. SubmitQuery (userId, QueryQId) .QueryQ could be a global query, which uses the community attributes, or a source query, which concerns one community member .
- The Query Solver takes as input the UserId, the QueryQ and the community definition C (userId, QueryQId, C) .It sends directly the query to the Queries Cache before its processing .The Queries Cache identifies the part of the query answer that exists inside it (userId, ProbQ, Qanswer),then return it to the Query Solver.
- If the answer satisfies most of the constraints expressed by the query then the result is delivered to the user
  Else the Query Solver continues the processing of the rest of the query. It rewrites Remainder Q the others parts that cannot be answered by the QC

(userId, Qnonexistant, C). The community will collaborate with peers to identify any external members who can answer this part of the query.

- Answers are collected by sub community then community and are added to the answers given by the Queries Cache then sent to QM which delivers the result to the user.

6.3 The Semantic Cache Mechanism

When a user formulates a query Q, Query Solver checks it first against the content of the semantic cache. The cache splits the user query into two portions, probe query and remainder query. The probe query Probe (Q) represents a partial answer to the query provided by the cache; it addresses those cache regions which contain communities' links satisfying the query and thus contribute to the answer. The remainder query Rem (Q) refers to data that should be shipped from communities.

If the remainder query is empty, the probe query serves the answer from the cache content and communities are not contacted. If the remainder query is not empty, its evaluation proceeds in a regular way. The answer to the user query is built up from the answers to the probe and remainder queries: Q=Prob (Q) $\cup$ Rem (Q).

For example, if the c ache contains the region R=``Patient AND Pathology", and query is Q=``Patient", the probe query coincides with R: Prob(Q)= ``Patient AND Pathology" , while the remainder query is Rem(Q)=``Patient AND NOT Pathology".

Once the cache detects some regions relevant to the query and constructs the probe query Probe (Q), the formula Q AND NOT Probe (Q) defines the remainder query Rem(Q). However, the following containment holds: Rem (Q)$\supset$ AND NOT Prob (Q)

In what follows we consider three operational cases processed by the semantic cache; each case refers to a particular relationship between a user query Q and regions in the cache. Below we list and describe all the cases. To process these cases the semantic cache takes in input cache with semantic regions and query Q and produces as output answer to Q and updated cache. It has to verify the query Q against all region descriptors in the cache:

- Equivalence: The cache contains a region R which formula is equivalent to the query formula: Q$\equiv$ R.
- Query containment: The cache contains one or more regions, which formulas R1, R2, Rm$\geq$ 1contains the query formula: Q $\subset$ R.
- Region containment: The cache contains regions R1, R2, Rm$\geq$1 which formulas are contained in the query formula: Ri $\subset$ Q.

7. Cache Maintenance

Web Services Communities are built by gathering Web Services related to a same field. However, as they evolve in a dynamic environment, communities could be affected by changes. A community is affected when one or more of its Web Services are affected. A Web Service is affected

when one ore more of its associated information sources are affected by schema changes.

- The above change can alter the semantic cache built among queries results collected from different communities. Hence, the semantic cache becomes stale and queries results stored in the cache become partially or totally undefined and lack the ability to satisfy users' requests.
- In this section we propose a maintenance process able to tackle the above problems by determining substitution for communities or Web Services affected by changes.

Our approach has the goal to preserve the maximum number of queries results affected by communities or Web Services schemas changes. The maintenance system is incorporated in the cache organizing the queries results by references of communities and Web Services that participated to answering queries.

Basics of our maintenance approach are the replacement of the affected community by one of its peers, and of Web Services according to replacement parameters.

As we have seen that communities in the CKB are related to each others via peer relationships according to peer constraints. Hence, communities affected by changes in the cache could be replaced by one of its peers. We propose here to add to the CKB Web Services replacement relationships according to replacement parameters in order to allow replacement of affected Web Services.

Replacement parameters	Semantics
$WS_1 \equiv WS_2$	WS_1 is equal to WS_2
$WS_1 \supseteq WS_2$	WS_1 is a super set of WS_2
$WS_1 \subseteq WS_2$	WS_1 is a sub set of WS_2
$WS_1 \approx WS_1$	Indifference

Fig. 3 Replacement Parameters

But before performing replacement, the following tasks must be ensured:

- The detection of the affected Web Services.
- The detection of the affected communities.
- The detection of affected Information Source.

8. Implementation

To evaluate our approach, we propose a prototype that is a Web Service based environment for building Web Services Communities.
In fact, our community support infrastructure supports building communities and allows communication between them and between user and communities.
In fact, Web Services [22] are emerging as promising technology for the effective automation of application integration across networks and organizations. They build upon XML, as vehicle for exchanging messages across applications, and upon the three major standards WSDL (Web Service Description Language), UDDI (Universal Description, Discovery and Integration),and SOAP, which provides the building blocks for service description, discovery, and interaction.
Java and Web Service technologies are suitable for the system implementation. We use a toolkit, the IBM Web Services Development Kit 5.0 (WSDK), which has supports for the Web Service protocols (e.g. UDDI, WSDL and SOAP). The toolkit also provides several components for developing Web services.
Concerning the proposed semantic cache mechanism, it is implemented in the Knowledge Broker (KB) system [23, 24].In fact, Knowledge Broker is a Web meta-searcher developed at the Xerox Research Centre Europe, it provides a uniform interface and query language for interrogating multiple, heterogeneous data repositories including standard databases (Oracle, Sybase, Informix etc.) and Web providers in different domains (newspapers, digital libraries, patents, medicine, etc).
The semantic cache is implemented in Java as an external service to the meta-searcher. It replaces the tuple-based cache mechanism used in the previous versions.
Service-like integration of the cache gives the main advantage of a clear separation between the query processing and cache processing. There are two main issues:

- It does not require any modification in the existing architecture of the system. The system is simply enhanced with the SemCacheManager module, which integrates the semantic cache into the system.
- The Web query system can decide whether to use the cache and which one (tuple-based or semantic one).

The semantic cache service is provided through an API with two methods, Request and Supply. The Request method takes a query with a set of destination sources and returns a remainder query for each source and the partial answer from the cache. The Supply method adds to the cache the answer tuples received from sources. The jobs in the cache (Request and Supply calls) are queued; a thread picks up the first one and processes it.

9. Conclusion and Future Works

In this paper we have discussed an infrastructure for managing and querying Web Services communities. Our approach focuses on offering a clear design of Web Services communities'. Further we proposed an optimization to the query processing based on using a semantic cache mechanism. We advocate the cache maintenance mechanism, in order to keep the availability and the consistency of the cache.
Our future work will be directed towards the management of the semantic cache space. Furthermore, we will focus on various problems of integrating semantically heterogeneous Web Services into communities.

References

[1] B. Benatallah, M. Dumas, Q.Z. Sheng and A.A.H. Ngu. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. 18th Int'l Conf, Data Eng, IEEE CS Press, pp. 297–308, California, 2002.
[2] G.Yan, W.Ng and E. Lim. Product schema integration for electronic commerce—synonym comparisons approach. IEEE Trans. on Knowledge and Data Eng, 2002.
[3] J. Ullman. Information integration using logical views. Theor. Comput. Sci, pp. 189–210, 2000.
[4] A. Y. Halevy.Answering queries using views: A survey. VLDB Journal 10 (4), pp. 270–294, 2001.
[5] M. Lenzerini. Data Integration: A Theoretical Perspective. In L. Popa (Ed.),PODS'02, pp. 233–246, Madison, Wisconsin, USA, 2002.
[6] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos and J. Widom.The tsimmis approach to mediation: Data models and languages. Journal of Intelligent Information Systems (JIIS) 8 (2), pp.117–132, 1997.
[7] A. Y. Levy, A. Rajaraman and J. Ordille. Querying heterogeneous information sources using source descriptions. In Proc. of VLDB, pp.251–262, Bombay, India, 1996.
[8] C. Beeri, A. Levy and M.-C. Rousset. Rewriting Queries Using Views in Description Logics. In. L. Yuan (Ed.), PODS'97, pp. 99–108, New York, USA, 1997.
[9] B. Benatallah, M.-S. Hacid, H. Paik, C. Rey and F. Toumani.Towards Semantic-driven, Flexible and Scalable Framework for Peering and Querying e-Catalog Communities. Information Systems Journal,Special issue on semantic web services, 2004
[10] K. Baina, B. Benatallah1, H. Paik, F. Toumani, C. Rey, A. Rutkowska and B. Harianto.WS-CatalogNet: An Infrastructure for Creating, Peering, and Querying

e-Catalog Communities. In Proc. of the 30th VLDB Conference, Toronto, Canada, 2004.

[11] H. Paik, B. Benatallah, and F. Toumani.WS-CatalogNet: Building Peer-to-Peer e-Catalog. In Proc. of 6th International Conference on Flexible Query Answering Systems, Lyon, France, June 2004.

[12] A. Crespo and H. Garcia-Molina. Routing Indices for Peer-to-Peer Systems. In.ICDCS'02, Vienna, Austria, 2002.

[13] P. Bernstein, F. Giunchigiloa, A. Kementsietsidis, J. Mylopoulos, L. Serafini and I. Zaihrayeu. Data Management for Peer-to-Peer Computing: A Vision. In WebDB'02, Madison, Wisconsin, 2002.

[14] V. Papadimos, D. Maier and K. Tufte. Distributed Query Processing and Catalogs for Peer-to-Peer Systems. In CIDR'03, Asilomar, CA, 2003.

[15] B. Yang and H. Garcia-Molina. Designing a Super-Peer Network. In ICDE'03, Bangalore, India, 2003.

[16] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst and A. Lser Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-To-Peer Networks. In WWW'03, Budapest, Hungary, 2003.

[17] M. Papazoglou, B. Kramer and J. Yang.Leveraging .Web-Services and Peer-to-Peer Networks.In CAiSE'03, Klagenfurt, Austria, 2003.

[18] H.-Y. Paik, B. Benatallah and F. Toumani. Towards Self-Organizing Service Communities.IEEE Transactions on Sys. Man and Cyb , 35(3) , pp. 408-419 ,2005.

[19] B. Benatallah, M.-S. Hacid, H.-Y. Paik, C. Rey, and F. Toumani. Peering and Querying e-Catalo Communities. Technical Report UNSW-CSE-TR-0319, CSE, UNSW, Sydney, Australia, 2003.

[20] M. J. Carey, M. J. Franklin, M. Livny and E. J. Shekita. Data Caching Tradeoffs in Client-Server DBMS Architectures. In Proc. 1991 SIGMOD Conference, pp. 357-366, 1991

[21] B. Chidlovskii and U. Borghoff. Signature File Methods for Semantic Query Caching. In Proc. of the 2nd European Conf. on Digital Libraries, LNCS 1513, Crete, Greece, September 1998.

[22] F. Casati, M.-C. Shan, D. G. (editors).The VLDB Journal: Special Issue on E-Services. 10(1), Springer-Verlag, Berlin, Heidelberg, 2001.

[23] J.-M. Andreoli, U. Borghoff and R. Pareschi. Constraint-Based Knowledge Broker Model: Semantics, Implementation and Analysis. Journal of Symbolic Computation,Vol. 21, No.4, pp. 635-667, 1996.

[24] J.-M. Andreoli, U. Borghoff, P.-Y. Chevalier and al. The Constraint-Based Knowledge Broker System. In Proc. 13th IEEE Int. Conf. on Data Engineering, 1997.

Hela Limam received the master degree in Computer Sciences applied to management Institute of management in 2007.She is actually teaching in the high institute of management of Tunis.