A semantic cache for queries optimization of Health care services communities

Hela Limam and Jalel Akaichi

Department of Computer Science, ISG, University of Tunis

hela.limam@isg.rnu.tn

Abstract. Services communities are currently one of the most important concepts for enabling effective communication between a potentially large numbers of distributed services. They often struggle to locate the relevant Web services needed to satisfy users' requests. This paper presents an approach for an efficient query processing of health care services communities. The proposed solution is based on endowing the community with a semantic cache. We show how the health care Web services community' members can considerably take advantage of the semantic cache while processing a query.

Keywords: Web services community, semantic description, health care.

1 Introduction

Despite their visible advantage and accessibility, the rapid growing number of published Web services prevents users or requestors from finding easily and efficiently the services relevant to their specific needs. Hence the concept of communities of Web services has emerged for gathering Web services according to their functionalities in order to ease and improve the process of Web services discovery in an open environment like the Internet. In fact, gathering Web services into communities aims to address complex users' needs that a single Web Service can not satisfy. Web services communities provide a centralized access to several functionally-equivalent Web services via a unique endpoint which enables the query processing.

These communities are built with the purpose to be queried transparently and easily by users, which aim to satisfy their informational needs in a satisfactory time and in a pertinent retrieval. In fact, a user query may involve the access of a number of distributed communities in order to locate Web services that are capable of answering the query . Those queries are sometimes complex and short-living. Hence, it seems to be beneficial to conserve them for a future reuse and in order to be shared by communities' members who have similar informational needs. In this context, enhancing Web services communities activities using a semantic cache memory highlights the interest of capitalizing formulated queries to the cache memory and in general the expert how-know of the community in the field of the information discovery. Hence, the process of researching resources for answering queries becomes based on a formal manipulation of annotated resources. In our paper we propose a model for Health care services a community which enables semantic caching of queries. We provide a formal description of queries and the cache content and we detail query processing inside a community using the semantic cache. The rest of this paper is organized as follows: In Section 2, we review previous research on semantic caching as well as other related issues. A Health care community model is defined in Section 3. Section 4 proposes a formal description of the semantic cache. Section 5 discusses the semantic cache organization while section 6 discusses semantic caching query processing strategies. The performance of semantic caching is examined through the system implementation in section 7 . Finally, we summarize our work and discuss future research in Section 8.

2 Related Works

A review of research projects aimed at assisting community activities by collective memory as the european project SevenPro [1], the projects ANR e-WOK HUB [2] and Immunosearch [3] or the projet C3R [4] highlight the need to capitalize requests made by users in suitable databases to allow their authors to reuse or exchange them with other community members. More generally, capitalization approaches to information retrieval becomes a real issue in many areas.

Authors in [5] propose an approach to clarify the critical procedures of information retrieval in the medical field using what they call the search strategies portals .Starting with a set of standard questions in the field, they define a set of patterns representing research procedures. A search procedure is represented by an ordered set of subgoals and for each search procedure, links to relevant sources of information are established

Authors [6] offer a browsing environment of Web resources. They distinguish among three levels of knowledge: Examples of research procedures for dynamic navigation systems also exist in online learning-based models of semantic Web. In [7], a model for the pedagogical approach is adopted by an assembly of requests parameterized and resources whose annotations meet these requests up educational material are presented dynamically to the learner navigating through the system.

In conclusion we can say that there has been much interest in the area of applying semantic caching in Web services and communities in general. Some of the proposed approaches only work in the field Web communities while others limit queries to Web services. Hence previous works either ignore the possibility of applying semantic cache for enhancing and performing query processing among communities of Web services. The objective of our work is to extend the existing semantic caching work along several dimensions. First we present a formal semantic caching model for Web services communities, and then we explore the semantic caching query processing strategies. We examine how to efficiently answer queries against the cache. Finally, we validate semantic cache performance through a detailed Health care case study.

3 Health care community model

Communities of web services are virtual spaces that can dynamically gather different Web s ervices h aving c omplementary functionalities i n o rder to p rovide c omposite services with high quality. Some approaches have been proposed to organize communities of Web services. In a previous work we proposed a Web services communities design language, called WSC-UML[8], which increases the expressiveness of UML for Web services communities and guides their design. Stereotypes and graphical annotations have been added to UML diagrams in order to distinguish between the different aspects in a Web services community.

In fact, the Health care community is designed to combine data from the large ancillary services, such as pharmacy, laboratory, and radiology, with various clinical care Services . IT has an identifier and is described by a s et of attributes. It is composed of a s et of Health care Web services: Insurance Service, Care Service, Patient Refferal S ervice, P hysician Refferal S ervice an d s cheduling Service. W eb s ervices insider a community are associated to each other's with peer relationships.

Each Web service modeled as a class in WSC-UML class diagram and is described b y a s et o f attributes as s hown in figure 1 . T he n umber o f i ntegrated W eb services involved in the Health care community is dependent upon the data structures and h as to p rovide a n in terface th at a llows c linicians to a ccess t he silo s ystems through a portal.

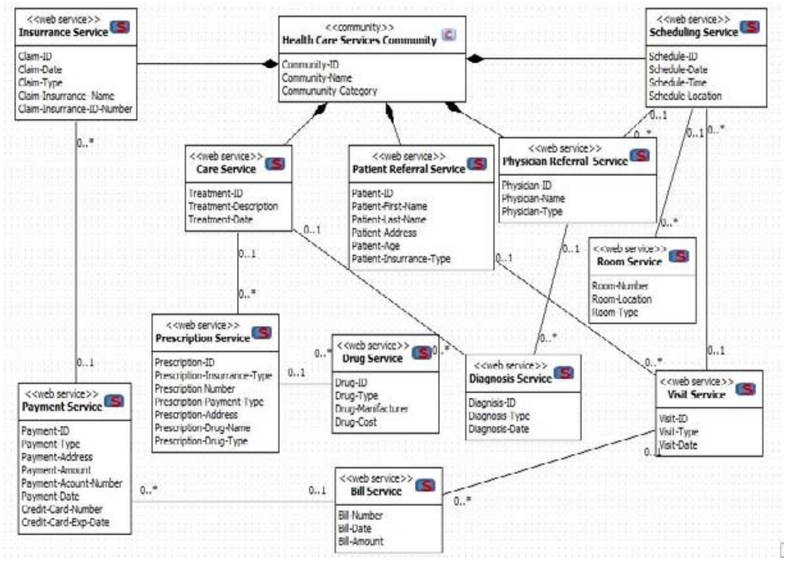


Fig. 1 Health care community model.

4 Formal description of semantic cache

Our proposed approach is a model for building, reusing and sharing of search queries for information and organizes them into formal research approaches also reusable and sharable inside a Web services community. We propose a model to formalize queries specifically, we focus on the construction and operation of a database query making operational stages of a scenario of information retrieval that we call the process of finding information. This knowledge base can be seen as episodic memories in which to research approaches are built dynamically based queries.

We consider that queries addressed to communities take the form of select and project queries, although the proposed model can be extended to handle other more complicated queries .the formal definition of the semantic cache is given in this section , later we discuss how to store and organize the semantic cache.

The Health care Web services Community modeled in the previous section is considered a relational database where each modeled class is a relation. For example the class Web service is a relation. Each relation consists of a relation schema and a relation instance.

An instance of the relation << Community Member>> is a set of members satisfying the constraint of having the same number of the attributes described in the Community Member schema. Suppose that the examined community C consists of a set of modeled classes $cl_1, cl_2,.., cl_n$

$$C=\{ cl_i , 1\leq i \leq n \}$$

Further let A_{cli} stand for the attribute set defined by the schema of the class cl_i and A present the attribute set of the whole community, then we have: $A= \cup A_{cli}, 1\leq i \leq n$.

Before defining a semantic cache, we first give the predicates, through which a semantic cache is constructed.

Definition 1:

Given a community $C=\{cl_i\}$ and its attributes set :

$A =\cup A_{cli}, 1\leq i \leq n$, a Compare Predicate of C, P, is of the form P=a op c, where $a\in A$, op $\in\{\leq, <, \geq,>,=\}$,c is domain value or a constant.

In fact, a semantic cache is used to store annotated results of queries. It is composed of a set of Semantic Query Results. A Semantic Query Result is an original, decomposed, or coalesced query result. Its definition is consistent with that of a materialized view [11]. To further simplify the problem, we assume that the selection condition of a query is an arbitrary constraint formula of compare predicates, namely, a disjunction of conjunctions of compare predicates.

Definition 2:

Given a community $C= \{cl_i\}$ and its attributes set $A=\cup A_{cli} ,1\leq i\leq n$.

A Semantic Query Result is a tuple $<C_C,S_A,S_P,S_C>$ where:

$S_C=\pi_{S_A}\sigma_{S_P}(C_C)$, $S_R \epsilon C$, $S_A \subseteq A_{CC}$ and $S_P = P_1 \cup P_2 \ldots \cup P_m$ where each P_j is a conjunctive of comparative predicates , i.e.,

$Pj=b_{j1} \wedge b_{j2} \wedge b_{jl}$,

Each b_{jl} is a compare predicate involving only the attributes in A_{CC}

In definition 2, C_C and S_A define the class and attributes involved in computing S, respectively, S_P indicates the select condition that the tuples in S satisfy. Hence, these three elements specify the semantic information associated with S. The actual content of S is represented by S_S. From the restrictions added, we can see that semantic query results are the results of Select-Project operations, with the selection conditions containing only compare predicates.

Before queries get answered, their contents are empty (i.e., $Q_C=\Phi$). Therefore, we formally define a query just as we define a semantic query result.

Definition 3:

A Query Q has the form $<Q_C,Q_A,Q_P,Q_S>$.

A semantic cache is defined as a set of semantic queries results. To reduce space overhead, the cached semantic queries results do not overlap with each other. In the following, we first give the concept of disjointed queries results, and then formally define a semantic cache.

Definition 4:

Two Semantic queries results $Si=<S_{iC},S_{iA},S_{iP},S_{iS}>$ and $Sj=<S_{jC},S_{jA},S_{jP},S_{jS}>$ are said to be disjointed if and only if :

1. $S_{iA} \cap S_{jA}=\Phi$ or

2. $S_{iP} \wedge S_{jP}$ is unsatisfiable.

Definition 5:

A Semantic Cache, SC, is defined as SC={semantic query result S_i} where $\forall$ j, k ($Sj \in SC \wedge S_k \in SC \wedge j \neq k$) S_j and S_k are disjointed).

5 Semantic cache organization

Several approaches tackle the problem of the physical storage of semantic queries results. The work in [10] stores queries results in tuples, and associates every query with a pointer to a linked list of the corresponding tuples. This approach works fine for select-only queries and memory caching. The key advantage is easy maintenance: tuples can be added, deleted, or moved between segments conveniently. However, this linked list scheme is not appropriate for disk caching, since it may result in too many I/O operations.

Moreover, when select-project queries are cached, the resulting tuples for different segments are no longer at the same length. Hence, even for memory caching, its advantage in maintenance is lost. Another noticeable disadvantage for this approach is the large space overhead caused by the tuple pointers.

In our case, semantic cache is composed of two parts: the content and the index. Every semantic query result is stored in one or multiple linked pages, and is associated with a pointer pointing to its first page in the memory (disk) cache. Each page contains a query result, rather than the community classes. The cache space is also managed at a page level, which makes semantic query result allocation and deallocation algorithms more straightforward and simpler. For allocation, if there are enough free pages to hold a query result, and then allocate the pages to it; for deallocation, just mark the deallocated pages as free. The index part maintains the semantic as well as physical storage information for every cached query result. In what follows, we list the basic items kept in the index. For every cached query result, we have:

- the name S, the community class CC, the attribute set SA, and the selection predicate SP ;
- the pointer pointing to the first page that stores the query result; and
- the timestamp indicating when the query result was last visited STS;

The index structure proposed here is consistent with the formal definition of the semantic cache. In addition to the four basic components of the semantic query result, we further add other items for maintenance use, such as STS. The semantic cache index is more clearly illustrated through the following

Example 1:

Consider the two classes: Patient referral service and scheduling services of the Health care community Patient referral service (Patient-ID, P address, P telephone, Pfirst-name,Plast-name,PAge,Pinsurance-Type) and Scheduling Services (Schedule-ID, Sdate, Stime,Slocation); and suppose that the cache contains four queries result:

S1: Select Plast-name From Patient referral service Where 20 < PAge < 60;

S2: Select Pfirst-name, Plast-name From Patient referral service Where PAge> 10;

S3: Select Schedule-ID From Scheduling Services Where SDate= 28/12/2010;

S4: Select Slocation From Scheduling Services Where Sdate > 10/12/2010 ;

Also, suppose that the first pages of the four queries results are one, three, five, and six, respectively, and Si was last visited at Ti, then the index is shown as Table 1.

Table 1: Queries results organization in the semantic cache

S	C_C	S_A	S_P	S_C	S_{TS}
S_1	Patient referral service	Plast-name	20<PAge<60	1	T_1
S_2	Patient referral service	Pfirst-name, Plast-name	PAge> 10	3	T_2
S_3	Scheduling Service	Schedule-ID	SDate=28/12/2010	5	T_3
S_4	Scheduling Service	Slocation	Sdate> 10/12/2010	6	T_4

6 Query processing using the Semantic cache

To process a query from a semantic cache, we first check whether it can be answered by the cache. If yes, the locally available results are computed directly from the cache. When the query can only be partially answered, we trim the original query by removing or annotating the already answered parts and send it to the database server for processing. In this section, algorithms for semantic caching query processing are examined.

6.1 Theoretic Foundation

From the concept of Derivability defined in [11] we introduce the following definitions.

Definition 6

Consider a semantic query result S<CC,SA,SP,SC> and a query Q= $<Q_C,Q_A,Q_P,Q_S>$,we say Q is answerable from S if there exist a relational algebra expression F containing only project and select operations, and only involving attributes in S_A, such that $F(S_C)\neq\Phi$ and $\forall t\ (t\in F(S_C)) \Rightarrow (t$ satisfies $Q_P \wedge t$ contains only attributes in $Q_A))$. Furthermore, if F (S_C) =QC, we say Q is fully answered from S; otherwise, we say Q is partially answered from S.

From Definition 6, we know that the key to compute a query from a cached segment is to find the function F, and to make sure that F can be executed on the segment. Sometimes, even the entire result of a query Q is contained in a segment S, Q still is not answerable from S.This is because some of the attributes needed in F cannot be found in S. So, in Definition 6, we add an additional restriction on F. The following Example 2 illustrates such a point

Example 2

Consider Health care community and the semantic cache described in Example 1. Consider a query:

Q = **Select** Pfirst-name, Plast-name

From Patient referral service

Where(PAge> 5)$\wedge$(Patient-insurrance-type= Personal).

Obviously, the result of Q is totally contained in S2, since every tuple which satisfies (PAge>10)$\wedge$(Patient-insurrance-type= Personal) will always satisfy (PAge > 5). However, Q cannot be computed from S_2, because we cannot find an F as specified in Definition 6.

Intuitively, Q seems to be computed from S_2 by a function

πPfirstNameσ(PAge>10)$\wedge$(PatientInsurranceType=Personal).

But the attribute"Patient-insurrance-type" used in this function is not in S_2 after the projection.

Definition 7:

Consider a query $Q=<Q_C,Q_A,Q_P,Q_S>$,the predicate attribute set, Q_{PA} , contains all the attributes that occur in Q_P , i.e.,

$Q_{PA} = \{ a | a$ is an attribute, and a occurs in $Q_P\}$.

Consider a semantic query result $S=<C_C,S_A,S_P,S_C>$, a query $Q=<Q_C,Q_A,Q_P,Q_C>$, and Q's predicate attribute set Q_{PA} . Then we have:

Statement 1

If $C_C=Q_C$, $S_A \cap Q_A \neq \Phi$, $Q_P \wedge S_P$ is satisfiable by C_C, and $Q_{PA} \subseteq S_A$, then Q is answerable from S.

Statement 2

If $C_C = Q_C$, $Q_A \subseteq S_A$, QP$\Rightarrow$ SP, and $Q_{PA} \subseteq S_A$, then, Q is fully answered from S.

Definition 8:

Consider a semantic segment S= $<C_C, S_A, S_P, S_C>$.

Let Y be the set of all attributes uniquely determined by the attributes in the attribute set X, with respect to S_P. If X $\subseteq S_A$, we say S is an extensible semantic segment, $S_A \cup Y$, denoted by S_A^+ is called the extended attribute set of S, and the semantic query result $S^+=<C_C,S_A,S_P,S_C>$ is called the extended query result of S.

Since S_A is uniquely determined by X with respect to S_P, if a tuple consisting of attributes in X satisfies S_P, when extended to contain attributes in S_A^+, it will also satisfy S_P. This makes it possible to extend S to S^+.

Notice that for each extensible query result, there could exist multiple different extended attribute sets, and hence multiple different extended queries results. To investigate how to use extended segments in query processing, we examine Example 2 again. Suppose "Pfirst-name" is the key of "Patient referral service" class, thus other attributes of "Patient referral service" can be uniquely determined by "Pfirst-name" Hence, S_2 is an extensible query result.

Clearly, "Patient-insurrance-type" can be uniquely determined by "Pfirst-name" To form S_2^+, we retrieve the tuples containing both "Pfirst-name" and "Patient-insurrance-type" from the community knowledge base appends them to S_{2C} according to the value of "Pfirst-name". After that, Q can be computed from S_2^+therefore, we have:

Statement 3

Consider an extensible query result $<C_C, S_A, S_P, S_C>$ and a query $<Q_C,Q_A,Q_P,Q_S>$

Suppose S_A^+ is an extended attribute set of S, and S^+ is the extended query result of S with respect to S_A^+, Q_{PA} is Q's predicate attribute set. Then, we have:

1. If $C_C = Q_C$, $S_A^+ \cap Q_A \pm \Phi$, $Q_P \wedge S_P$ is satisfiable, and $Q_{PA} \subseteq SA^+$, then Q is answerable by S^+.
2. If $C_C = Q_C$, $Q_A \subseteq S_A$, $Q_P \Rightarrow S_P$, and $Q_{PA} \subseteq S_A^+$, then Q can be fully answered by S^+.

Definition 9:

Consider a semantic segment $<C_C,S_A,S_P,S_C>$ and suppose K_A is the primary key of C_C. If $K_A \subseteq S_A$, we say S is a key-contained query result.

Definition 10:

Consider a key-contained query result $S =<C_C,S_A,S_P,SC>$, and a query $<Q_C,Q_A,Q_P,Q_S>$

Suppose Q_{PA} is its predicate attribute set. Then, we have:

1. If $C_C=Q_C$, $Q_P \wedge S_P$ is satisfiable, then Q is answerable by $S^+<C_C, S_A \cup Q_A \cup Q_{PA}, S_P, S_C^+>$.
2. If $C_C=Q_C$, $Q_P \Rightarrow S_P$, then Q can be fully answered by $S^+ = <C_C, S_A \cup Q_A \cup Q_{PA}, S_P, S_C^+>$.

6.2 Query processing

Since a community does not store Web services locally, processing the query requires locating Web services that are capable of answering the query. These Web services can be selected from the local members of the community or from the semantic cache. We propose a collaborative query processing technique that consists of two steps:

(i) Dividing the query into parts when put together, satisfy all constraints expressed in the query, (ii) resolve the query by sending it to the selected parts.

We adopt a query rewriting algorithm, which takes as input the community classes ,S<C_C,S_A,S_P,S_C> and the query Q<Q_C,Q_A,Q_P,Q_S>then produces the following output:

Qlocal: the part of the query Q that can be answered by the community's local semantic queries results S, that is, the attributes specified in the queries that are supported by the local members. It also gives the combination of the local members that can answer all (or part of) the query.

Qrest: the part of the query that cannot be answered by the local queries results. The community will identify any external members who can answer this part of the query. Hence, Qrest is forwarded to peers. The expected answer of the forwarding is the combination of the external members that are capable of answering Qrest. Relationship between Q and S fall into five types as described in figure 2

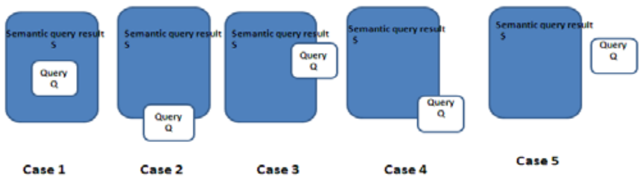


Fig. 2 Types of relationship between Q and S

To summarize the query rewriting we present Query Rewriting Algorithm, this rewrites a query Q via a key-contained semantic query result S which is defined on the same relation as Q.

Query Rewriting Algorithm

QueryRewrit (Query Q, Seamtic query result S, Query lq, Query aq, Query rq1, Query rq2, int type),to rewrite a query Q via a semantic query result S.

Input: Query Q; key-contained semantic query result S

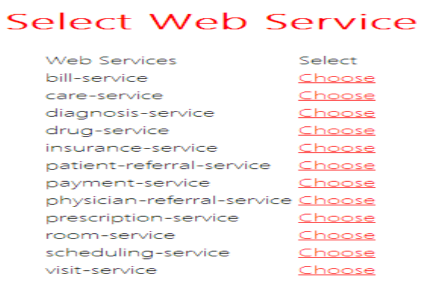
Output: local Query lq; Amending Query aq; Rest Query rq1, rq2; Tr Type type

```
 Procedure: {
$K_A \leftarrow$S's key attribute set;
$A_1 \leftarrow (Q_A \cap S_A) \cup K_A$;$A_2 \leftarrow (Q_A - S_A) \cup K_A$;
IF $Q_A \subseteq S_A${
IF $Q_P \Rightarrow S_P$ {
/***** Case 1*****/
type = 1;
IF ($Q_{PA} \subseteq S_A$) THEN aq = NULL; ELSE
aq = $\pi K_A \sigma Q_P Q_R$);
lq = $\pi Q_A \cup K_A \sigma Q_P (S_C)$
```

```
    rq1 = rq2 = NULL; return; }
    IF (QP∧SP is satisfiable) {
    /***** Case 2*****/
    type = 2;
    IF ($Q_{PA} \subseteq S_A$) THEN aq = NULL; ELSE
    aq = $\pi K_A \sigma\ Q_P \wedge S_P\ (Q_R)$;
    lq = $\pi Q_A \pi \cup K_A \sigma Q_P(S_C)$;
    rq1 = $\pi Q_A \cup K_A\ \sigma\ QP\ \wedge\neg S_P\ (Q_R)$;
    rq2 = NULL; return; } }
    IF ($Q_A \subseteq S_A$)does not hold {
    IF ($Q_P \Rightarrow S_P$ ){
    /***** Case 3 *****/
    type = 3; lq =$\pi\ A_1(S_C)$; rq1 =$\pi\ A_2\ \sigma Q_P(Q_R)$;
    rq2 = aq = NULL; return; }
    IF ($Q_P \wedge S_P$ is satisfiable) {
    /***** Case 4 *****/
    type = 4;
    lq = $\pi A_1(S_C)$;
     rq1 = $\pi Q_A \pi K_A\ \sigma\ Q_P\ \wedge\neg S_P\ (Q_R)$
    rq2 = $\pi A_2 \sigma Q_P \wedge S_P(Q_R)$; aq = NULL; return; } }
    /***** Case 5 *****/
rq1 = Q; \ pq = aq = rq2 = NULL; type = 5; return; }
```

7 Experimental evaluation

The developed application works as described in next steps:
We first choose the service to which we address the query as chows in Figure 3

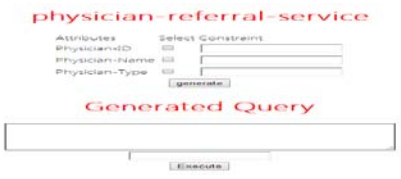


Select Web Service

Web Services	Select
bill-service	Choose
care-service	Choose
diagnosis-service	Choose
drug-service	Choose
insurance-service	Choose
patient-referral-service	Choose
payment-service	Choose
physician-referral-service	Choose
prescription-service	Choose
room-service	Choose
scheduling-service	Choose
visit-service	Choose

Fig.3 List of Web services

Then, a new page is loaded in which we have all service attributes (fig.4)

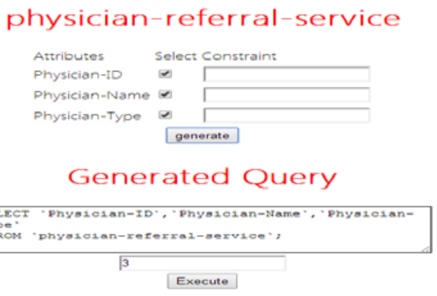


physician-referral-service

Attributes	Select	Constraint
Physician-ID	☐	
Physician-Name	☐	
Physician-Type	☐	

Fig.4 Services attributes

We choose the attributes and constraints, the system automatically generate the query when clicking on the button (fig.5)



physician-referral-service

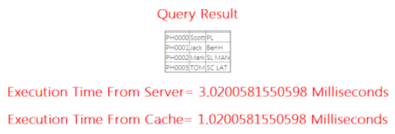
Attributes	Select	Constraint
Physician-ID	✓	
Physician-Name	✓	
Physician-Type	✓	

```
SELECT `Physician-ID`,`Physician-Name`,`Physician-Type`
 FROM `physician-referral-service`;
```

3

Fig.5 Query generation

We click on the execute button in order to get query result from the server and from the cache if it already exists (fig.6)



Query Result

PH0000	Scott	PL
PH0001	Jack	BenH
PH0002	Mark	SL MAN
PH0003	TOM	SC LAT

Execution Time From Server= 3.0200581550598 Milliseconds

Execution Time From Cache= 1.0200581550598 Milliseconds

Fig.6 Execution time

Based on results given by the application, we tested two types of queries, for each one, we compare the execution time both on server and from Cache.

Simple QUERY

```
SELECT * FROM Physician-referral service
```

We have launched a script that randomly inserts 20.000 records on this table; we have compared execution time while increasing the records number. The whole query is executed via web service. We got the following results (fg.7):

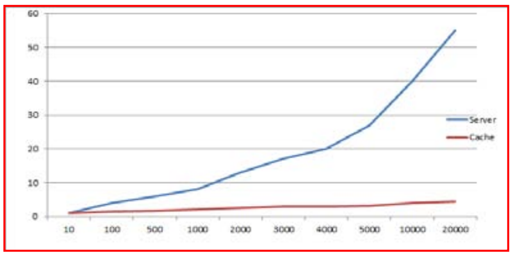


Fig.7 Execution time for a simple query After the simple query, we tested the prototype with a join query

Join Query

```
SELECT payment-ID, payment-type, Bill-date FROM
Payment-service, Bill-service WHERE payment-
service.Bill-Number=Bill-service.Bill-number ;
```

This query is more complex and it need more time and space, in this case the use of cache for storing query results if more useful and results have proven this theory

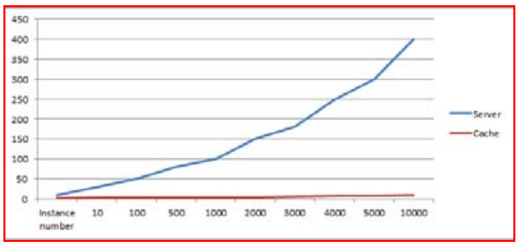


Fig.8 Execution time for a join query

8 Conclusion and future works

We presented a semantic cache mechanism designed for enhancing querying a Health care community. Query caching allows reusing answers to previous queries, so reducing the delivery time of answers and the traffic on the Net.

Semantic cac hing i s b ased o n t he semantic r epresentation o f cac hed d ata an d processing queries by construction of local queries for retrieving cached data and rest queries f or f etching d ata f rom r emote s ervers. W e p roposed cach e ar chitecture f or caching multiple q ueries ad dressed t o the co mmunity a nd considered al l o perational cases. F or all t ypes of answers we have developed al gorithms for q uery evaluation against t he cache content. Our f uture work i ncludes a large te sting o f the s emantic cache mechanism i n the presence of a r eal user request. W e al so p lan to propose re-placement s trategies f or t he cach e maintenance i n o rder t o t une t hem b etter to r eal user p rofiles. Our i mportant r esearch issue i s the extent of o ur s emantic cac he to a more powerful query language such as SPARQL.

References

1. Cherif H,Corby O, Faron C, Khelif K,: Semantic annotation of texts with RDF graph contexts. In : Proceeding of t he I nternational C onference on C onceptual Structures (ICCS'2008), pp.75 -82 (2008)
2. Khalid B elhajjame , M athieu d'Aquin , Peter H aase , Paolo M issier,: Semantic hubs for geographical p rojects". I n : Proceeding o f S emantic M etadata M anagement an d A pplications (SeMMA),workshop at ESWC, pp. 3–17 (2008)
3. Kefi L.,Demarkez M,Collard M, : A knowledge base approach for genomics data analysis. In : Proceeding o f t he I nternational C onference o n S emantic Systems, G raz, A ustria, (2008)
4. [4] Faron C,Mirbell I,Sall B, Zarli, : Une approche ontologique pour formaliser la connaissance ex perte d ans l e modèle d u contrôle de conformité en construction .In : 19ème j ournées f rancophones d 'ingénierie d es co nnaissances, N ancy, F rance,Capaudes E ditions (2008)
5. BhavnaniS, B ichakjian C, J honsonT, Little R, Peck F, S trecher V, : Strategy hubs : N ext generation domain protals with search procedures. In: Proceeding of ACM Conference on Human Factors in Computing Systems, pp. 393–400, Florida, USA (2003)
7. Buffereau B, Duchet P, Picouet P,: Generating guided tours to facilitate learning from a set of indexed r esources. I n : P roceeding of I EEE I nternational C onference on A dvanced Learning Technologies (ICALT), pp. 492, Athens, Greece: IEEE Computer Society (2003)
8. Yessad A, F aron C, D ieng R, L ASKRI M,: Ontology-driven ad aptive co urse g eneration for web-based ed ucation. In : World Conference o n E ducational Multimedia, Hypermedia andTelecommunications(ED-MEDIA),Vienna, Austria (2008)
9. Limam H ela, Akaichi J alel, O ueslati W ided: WSC-UML: A U ML P rofile for M odeling Web S ervices C ommunities: A Health C are C ase S tudy.In: International J ournal of A dvanced Research in Computer Science, Vol.2, No.2 (2011)
10. A. Gupta , I. Singh Mumick,: Maintenance of Materialized Views: Problems, Techniques, and Applications, Data Eng. Bull.,vol. 18, no. 2, pp. 3-18 (1995)
11. S. D ar, M.J. Franklin, B .T. J onsson, D. S rivatava, M . T an,: Semantic D ata C aching a nd Replacement. In : Proceeding of VLDB Conf.,pp. 330-341 (1996)

12. P.A. Larson, H.Z. Yang,: Computing Queries from Derived Relations. In: Proceeding of Very Large Databases, pp. 259-269 (1985)