International Journal of Computer Applications (0975 – 8887)
Volume 43– No.23, April 2012

Semantic based Efficient Cache Mechanism for Database Query Optimization

P.Mohan Kumar
Assistant Professor(Sr)
School Of Information Technology
Vit Vellore,India

J.Vaideeswaran
Senior Professor.
School Of Computing Sciences
Vit Vellore India

ABSTRACT

The primary goal of the database system is to provide the user a convenient and efficient access to the query related data. With this concern this paper provides semantic based cache mechanism techniques for optimizing the user queries. Here the frame work for optimization is analyzed which supports data and computation reuse, query scheduling and cache efficient utilization algorithm is presented in order to improve the evaluation process and minimize the overall response time. Further the case study is analyzed to test the performance and extended the same for multi-queries to achieve parallelism.

Key words

semantic, cache efficiency, optimization.

1. INTRODUCTION

Query optimization in database as received lot of attention. As the data set sizes continues to grow its increasingly important and challenging to execute. However the basic principle of database is to provide the user a convenient and efficient access to the specified query irrespective of database types, with this concern we concentrate user convince by providing the user a semantic base information relevant to their query with that they can be able to get the exact as well as nearby information's and cache efficiency mechanism to reduce the latency time for data access as system side thus the optimization overall. The main objective is to describe the query processing system that handles the semantic cache i.e. the concept of cache is introduced to retrieve the data efficiently, reusing the already retrieved data and using it against the similar user specified queries. Semantic cache mechanism is widely studied for both SQL and XML queries however we concentrate on SQL on at present for our aspect. As specified in [1] the methodology for storing the semantic and handling the cache for updating and reusing it as well the managing the space by replacement based on query is deployed. But in some cases as specified in [2] page replacement plays optimism here the cache page is segmented with respect the query segmentation and then the replacement policy is employed. For processing the semantic query the methodology specified in [2] is followed. The processing architecture for preprocessing and storing the resultant queries as per [3] is analyzed and for the partial query matching the approach specified in [3] is considered. The processing algorithm limits to selection and projection only we extend for join and nested queries. The existing system as a case study is analyzed and the algorithm is modified in order to achieve the problem statement by covering the basic database semantic query problems as specified in [4] is considered and future extension as multi-queries processing to achieve parallelism and conclusion we presented in rest of the paper.

2. SEMANTIC CACHE SYSTEM.

The issues related to cache system as detailed in [5] is analyzed firstly as where to install the semantic cache either to install in a separate proxy server through which all clients connects to server or to implement cache on server side the reduces the load of middle ware (web server).as shown in Figure: 1 and Figure: 2 .Secondly for what semantic should be defined and how to define it based on what constraints thirdly how to map it to the user based incoming query in a convenient manner as stated in [6]. With respect to cache management similar in [7] is to determine which data item must be retained in cache which one should be replaced to make free space for new data with given limited space for cache. The proposed work gives solutions for all these issues.

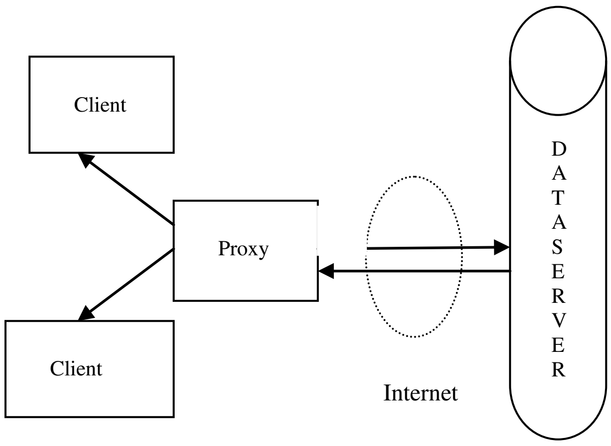

Fig 1: Cache with proxy server

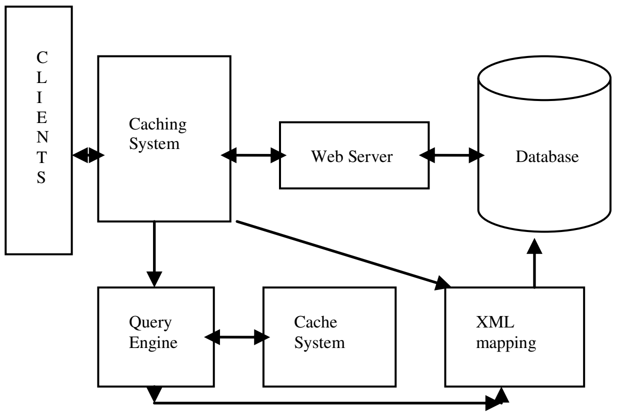

Fig 2: Cache with web server.

3. PROPOSED WORK.

3.1 Based on the architecture specified below the query is processed.

The semantics for all the possibilities related to the database can be defined if necessary for each semantic a predicate notation can be mentioned as specified in [2].In case of homogeneous schema set of all possible queries can also be defined and stored as a semantic reference. Set of rules were defined with respect to the operators available in suitable database language preferred. These can be used for mapping purpose during query execution. Predicates are matched based on the predicate values.

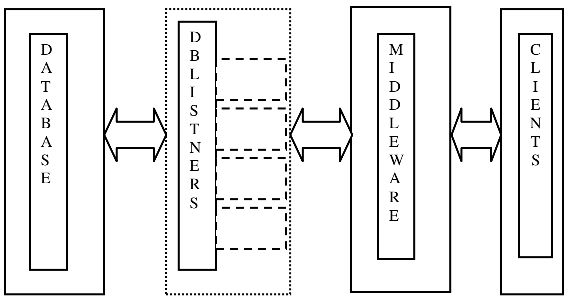

Fig 3: Proposed work architecture.

4. GENERALISED ALGORITHM

```
Input -user incoming query.
Output- result.
{
 For (each query perform the following steps)
    1. Check the table (user query relation) is available
    2. If available in data-cache then do
                    Lexical analysis (); Syntax analysis ();
            Get metadata value and apply one of the below operation
                    Exact match ();
         Partial match ();
            Mismatch ();
            Projection ();
         Go to step 5;
    3. If not available in data-cache then
              Invoke database server table();
                   Query the corresponding database;
              Return the result;
    4. Update the data to the data-cache // check for redundancy and null values.
    5. Return ();//return result to user.   }
```

Middleware architecture as shown in Figure 3 is designed based on the algorithm design. As specified in [1] the query is processed by splitting the user query into segments as selection, relation and conditional case and processed with query matching technique and then the query is rewritten if the query is partially related and rejected if doesn't match. But the process is limited to only selection and projection. In case of Join queries the query is decomposed in to sub query and then processed individually. For this the dynamic hashing technique is used to get the final result. Here the fact table is employed in order to store the details of sub-query referencing. The process architecture is shown in Figure 4 and detailed explanation in [3].both the approach is considered for our proposed technique since it extends for multi query processing. the user perceivable response time is summed up as (Ttotal = Tqueryshipping + Tqueryexecution + Tresultshipping)

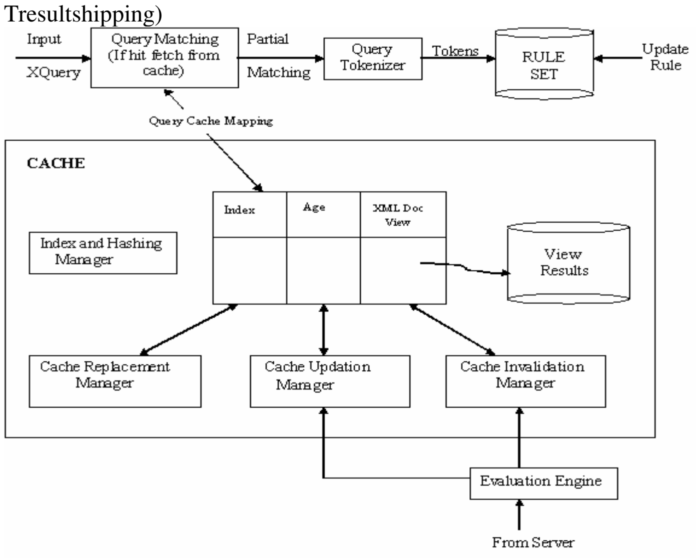

Fig 4: Detailed processing model.

5. CASE STUDY.

In this section the proposed approach is analyzed with the Fully Flexible Credit System model. This is a university based database which provides students to register their subjects related to their required credits by choosing the timeslot, subject, and faculty on their preference. The information can be accessed in distributed manner. The updating is done only at server side and the database access is streamlined by providing a middle ware and database listeners with secured manner. the user authentication is performed by load balancing by limiting the resource to certain level. In order to improve the optimism among user as well system the semantic cache mechanism is proposed to this system. In case if student wish to register for particular subject with particular time under particular faculty if valid status exist immediately his information is provided. In case in requirement is not available immediately it's rejected. for this purpose the query no need to get executed by server this can be done by referencing the cache itself since the processed result is stored in it. In case if the student specifies a query with partial then the information matches part in cache then its similar details were processed and issued to the user so that he can do his application correctly. Since the student registration limits to semester wise all possible query can be analyzed and stored in semantic cache with predicate notations so the it will be more useful for user level as well system response time.

5.1 Example to highlight the problem

Here we assume that user and semantic predicates are given as below:
PAS = age>40 Λ Sal>50k and
PAU = age<50 Λ Sal<60k
In this example algorithm will return yes. Meanwhile if users pose queries with different attributes like:
PAU = eName = "Komal"
Then algorithm will return no because PAS '# P AU. Even some of the data may exist in cache with following predicate:
(age>40 Λ Sal>50k) Λ (eName = Komal)
Thus the query processing performed by matching, partial matching, mismatching .These can be extended for multiple queries and parallelism can be achieved.

5.1.1. Work flow representation

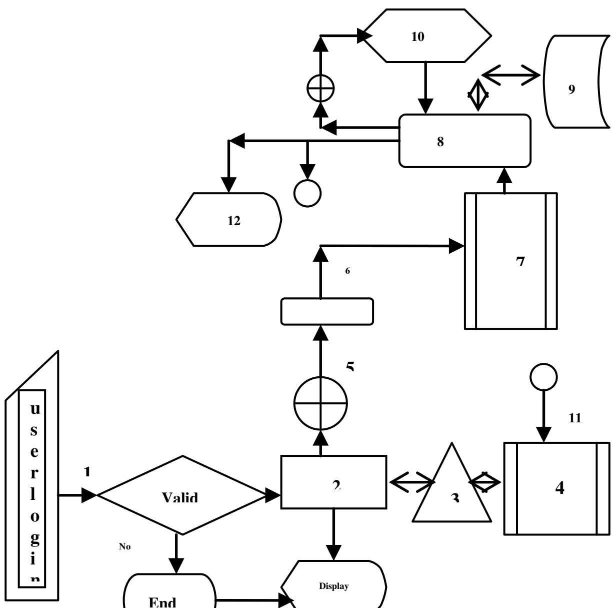

Fig 5: Work flow diagram.

The detail how case study works is shown in Figure: 5 and it maps the proposed algorithm implementation. The setup is as follows, here we use three methods as sequence choice to improve the performance one matching the stored results, two query matching as per [8] and thirdly normal query processing and storing the processed valid result for future use. Here we used the standard flow notations. Step 1. is the user login, as e.g. register no: 09bit120 if its valid then it will be processed (step 2) i.e. parsing the value as year, course and the required data from the semantic cache corresponding course details is filtered (step 3).Based on these the user can write a query .if the query related to preprocessed one then the required data is displayed (step 4) and (step 5.)If not it process (step 6)checks for semantic query stored in (7) possible queries are analyzed and stores as semantic cache (for limited application in our case)as stated in[9] and processed based on the query matching methodology then (8) extract the required data from stored database(9) as sample shown in table 1 and displays the output(11) else (step 10).If the query is new the it will be processed by the query evaluation engine with normal process(SQL server based)and displays the output (step 12) and update the processed query in (4)as shown in table 2 by details in[10]. Here the major issue is how to shift from one choice to other, how to manage if more than one user request arise and how to manage the cache coherence and in-memory process during execution and displaying the processed result. Thus the query processing phases are minimized depending upon the user input information by providing semantic cache as well managing the cache coherence for multi users thus overall the processing is optimized and performance is improved both system side as well as application point of view. The difficulty depends upon the user query so the utmost important is given to query processing and if query consists multi operators such has join in such case the process management as shown in detail processing model and the methodology as discussed in [3] similarity is used. In case of multi queries to distribute and group the result among the processor is done by the methodology below shown is figure 6.

Table 1. Sample semantics.

Sno	Semantics	Meaning
1.	BIT	Bachelor of Information Technology
2.	BIO	Bachelor of Bioinformatics
3.	BME	Bachelor of Mechatronics
4.	OS	Operating System
5	DBMS	Data Base Management System

Table 2: Sample Query Semantics

Sno	Sample Query	Semantic Query
1.	Select eage from employee where eage> =30	Select eage from employee where eage between 30 and 45.
2.	Select Fname from emp where subject ='OS'	Select Fname from emp where fname ='ram'and subject ='OS'
3.	Select cust_no from loan,account where account.cno=loan.cno and bankname='SBH'	Select cust_co from customer and bname='SBH'.//list holds those who holds account as well as loan.

Table 3.Sample Stored Outputs

Sno	Subjects	Slot	Faculty	Register no
1	DBMS	B1	RAM	111200
2	OS	B2	SAM	111201

6. MULTIQUERY MECHANISM.

In case if the user specifies multi query then its processed as shown in Figure 6 .Here a monitor routine set up is made at server admin level to manage resource utilization in conjunction with parallel execution environments. The algorithms specified for single query can be used in routine to achieve parallelism. The user query will be decomposed into sublevels and each query is processed as individual query and their results are cached, a monitor program takes care for checking the final result updating.

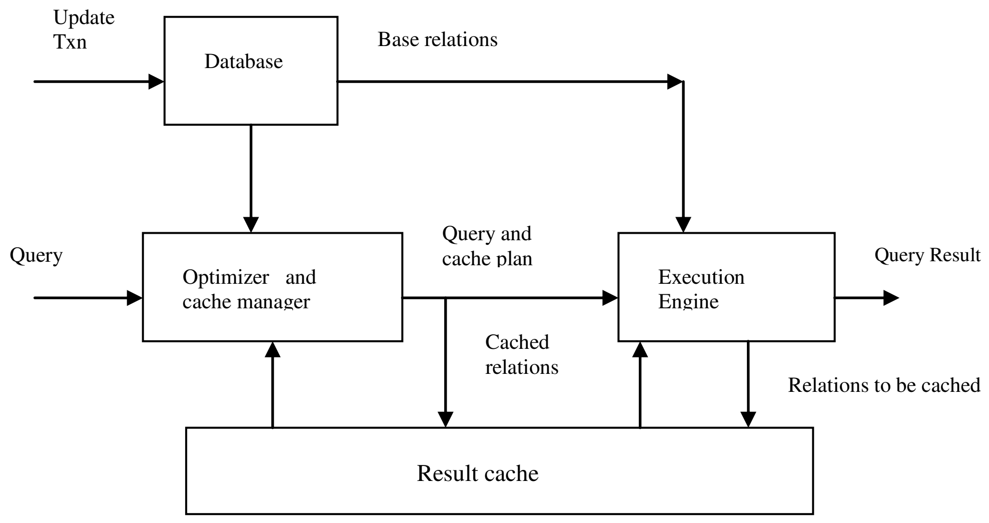

Fig: 6-Query result caching representation

7. CONCLUSION

In this paper we concentrated with an approach for primary goal of database that is to provide user a convenient and efficient way to access the required data. This can be achieved by proper and best processing of query. Our paper provide a way to process a user query by providing user a semantic information related to query ,so that the processing time can be minimsed, more over it allows to process partial information also. The analysis of mapping semantics to exact and partial query is discussed and to speed up the processing time we use cache efficiency mechanism (for prefetching and replicating the required data) that is details of database prior process and to store the processed result, for this an efficient query matching algorithm is designed, and implemented .The case study is made to test the presented algorithm as well as an approach is analyzed for multi query processing to encapsulates parallelism. This can be extended for distributed applications by using the dynamics cache managing technique at server and client level .Further this can be used for
real time applications in an optimistic way by using various semantic web tool such as RDF, OWL, SWRL, SPARQL as specified in [11].Thus over all as conclusion the paper provides a way for security, easy access, optimistic query processing with help of semantic cache mechanism to achieve the goal.

8. ACKNOWLEDGEMENTS.
I thank my Guide Dr.J Vaideeswaran for his invaluable guidance who keeps patience while explaining and gives a boon ideas and motivation and I thank our university which made us interest in research as well providing no time concerned laboratory and library facilities and my kid face and talk which is the grace to me.

9. REFERENCES

[1] S.Prabha, A.Kannan Anna University 2006 "An optimizing query processor with an efficient caching mechanism for distributed databases". International Arab journal of Information Technology Vol 3 July.

[2]. Munir Ahmad, Abdul quadar MD Ali Jinna university Pakistan 2005 " A efficient query matching algorithm for relational data cach*e* "Intelligence cache management for grid Australia.

[3] Sumalatha A.Vaidhei 2007 "XML query processing-semantic cache mechanism "IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.4, April.

[4] Min Wang, Haixun Wang University of Hawaii China 2009 Semantic queries in database problems and challenges. *CIKM'09,* November 2–6, Hong Kong, China.

[5]. Brian D. Davison Department of Computer Science Rutgers, the State University of New Jersey (USA) A Web Caching Primer c IEEE. Reprinted from IEEE Internet mputing, Volume 5, Number 4, July/August 2001, pages 38-45.

[6]. Research challenges and perspectives of the Semantic Web Report of the EU-NSF strategic workshop held at Sophia-Antipolis, France, October 3rd.

[7]. Q. Yang, H. H. Zhang, and T. Li. Mining web logs for prediction models in WWW caching and perfecting.

[8]. Qiongluo, Naughton, Sekar "Active Query Caching for Database Web Servers"

[9]. Abdullah Balamash and Marwan Krunz Performance Analysis of a Client-Side Caching/ Prefetching System for Web Traffic National Science Foundation through grant ANI-0095626.

[10]. M. Rabinovich and O. Spats check. Web Caching and Replicatio*n*. Addison Wesley, 1st edition, December 2001.

[11]. John Hebeler,Ryan Blace Semantic Web Programming Wiley India Edition ISBN 978-81-265-2110-4 pp-519-555.