Chapter

A Query Matching Approach for Object Relational Databases Over Semantic Cache

Hafiz Muhammad Faisal, Muhammad Ali Tariq, Atta-ur-Rahman, Anas Alghamdi and Nawaf Alowain

Abstract

The acceptance of object relational database has grown in recent years; however, their response time is a big concern. Especially, when large data are retrieved frequently on such databases from diverse servers, response time becomes alarming. Different techniques have been investigated to reduce the response time, and cache is among such techniques. Cache has three variants, namely tuple cache, page cache, and semantic cache. Semantic cache is more efficient compared to others due to capability to store already processed data with its semantics.
A semantic cache stores data computed on demand rather than retrieved from the server. Several approaches proposed on relational databases over semantic caching but response time on relational database is unsatisfactory. Hence, we proposed object relational databases over semantic cache. It is a novelty because semantic cache is mature for evaluation of relational databases but not for object relational databases. In this research, the implementation of query matching on object relational database with semantic caching along with object query is investigated to reduce the response time. Then, a case study is conducted on an object relational database model, and an object (relational database) query with semantic segment is applied. Results depict significant improvement in query response time.

Keywords: semantic cache, query matching, probe query, remainder query, object relational query

1. Introduction

Data size increased day by day, due to large data response time is going slow. In this regard, according to [1], relational databases can be used due to their better response time. Its idea is based on distributed database, which is helpful to reduce the data load and make access easy. In several scenarios, it occurs that structure must be continuously modified in multiple respects due to change in data types [2]. A relation in database is made up of several relations corresponding to relational database schema. The objective of a relational database design is to create a set of relation schema that allows user to store information and to retrieve information easily [2]. Relational database is structured in table, fields, and records. Relational database also delivers relational operator to manipulate the information kept in the

database tables [3]. Most RDBMS use SQL as database query language. The relational data base model is an extensively used data model, and a huge majority of existing database systems are based on the relational model [2]. Dr. E.F. Codd, a mathematician and research scientist at IBM, designed the relational model. Although most of the current RDBMS are not aligned to the Codd's model, yet it is considered as RDBMS [4]. Mitigating data redundancy and enhancing data integrity are two design principles of Codd's model [5]. The relational model also defines several logical operations that could be performed over the data. The relational data model has established itself as the main data model for commercial data processing applications [4]. Its achievement in this area has led to its applications outside data processing in systems for computer aided design and other environments [6]. Various issues in efficiency arise in RDBMS such as lack of handling the advanced data type, a restricted set of built-in types that use only numbers and strings. Also, certain types of relationships between database objects are hard to represent in the relational database model [7]. The RDBMS is pretty good in handling most information problems. But for new type of data type's problems, RDBMS technology could be superior upon. So, to attain the deficiency of RDBMS, we move on to ORDBMS [3]. To conform to the SQL standards, relational database model is reconsidered by ORDBMS; however, the object relational data model is a new definition altogether [7]. Object oriented languages like Java and C# can be integrated to ORDBMS, pertaining to their class, method, and objects features that are useful for the programmers for better integration. Moreover, ORDBMS schemas have additional features compared to its earlier counterpart [8]. ORDBMS model supports the object oriented features like abstraction, polymorphism, inheritance, etc. [9]. Like RDBMS, ORDBMSs rely on SQL queries and declarative approach for accessing and manipulating data rather than a procedural approach [10]. Occasionally, a mismatch in the programming language structure (procedural, declarative, or functional) and ORDBMS engine may occur at the time of database connectivity and access, which may results in performance issues. In a architectural point of view, ORDBMS is different than OODBMS that use a distributed approach while the former uses a centralized approach [8]. Nevertheless, this issue can be resolved by replicating ORDBMS over several machines. The further significant outcome of the technology is that it makes it possible to build information systems to address data management problems that are usually considered to be too challenging. In terms of interoperability, ORDBMS has two benefits. First is compatibility with the existing RDBMS components and second an object oriented access for the users and programmers. Similarly, storage and access mechanism, query processing, and optimization are the significant challenges in employing ORDBMS [7]. One of the efficient ways to mature a very large database is to distribute it between various server nodes [11]. Now ORDBMS is going to convert into the semantic caching and query result is more efficient in the semantic cache. Semantic cache is used to response the query in part such as Probe and Remainder [1]. Some part of query is answered from the cache and some is from the server [4]. ORDMBS is used in this technique in an intelligent way to answer the query result [8]. Different methodologies are easily making on semantic cache, and different definition of semantic cache content is used. Semantic cache content is {Rs, As, Ps, Cs}. Result is accessed according to the cache content, and a complete set of cache part is known as cache [1]. Suppose a user gives a query SELECT student id, name FROM student WHERE age < 28. In this query, Student ID and Name are answered from Probe part (saved in cache) but for Name, where clause it will be accessed by the remainder part (server result). So, it will consume more time if query is complex and more result is accessed from server [1]. For that ORDBMS is mainly focused on sematic caching but query optimizing technique is used, and this technique is going to be improved

by Query Matching part of query optimizing. For more understanding, we take the help from the problem statement.

- Query matching technique on ORDBMS is hard to implement.
- Query retrieving is time consuming.
- Query matching approach is unsatisfied on complex data.

The objective of this research is to overcome the said issues. In this regard, semantic cache query matching technique is proposed to extract useful contents from the cache to improve query response time. Then, query matching is investigated for object relational query on complex data by exploiting the semantic cache, and results of object query and relational database query are compared. In this section we take discussion for our proposed approach for query matching on object relational database query over semantic cache. Query matching approach is satisfying the query result with the object relational semantic content. Our projected approach being employed is on semantic cache architecture with object query and with the below concept. Firstly, the query is accessed on the complex structure and data set, so object relational query is used with row reference to access the query [7]. Secondly, the relation in database is retrieved with the object query but in the form of object [10]. The query is into the part by using query semantic content of QS, QF, and QS and result is retrieved according to adaptive region of semantics segments [1]. The study is mainly focused to indicate various advantages of semantic caching and then the simple workloads where the indication also includes the low overheads, decreased amount of network traffic, physical layout of database that is insensitive, and additionally a source to minimize and answer the queries without the participation of server [4]. In addition, handling the complexity of the workloads and depth coding for queries is left for quick processing query at server. By manifesting the semantic caching works on object database with usage of complex workloads, we would investigate the wide variety of applications particularly in an environment that is network constrained [12]. Semantic cache plays an important role in fetching results. Semantic cache is divided into two main parts. One part that is answered with the help of cache is the Probe query and the second part that is answered from server is the Remainder query. Query when passed to on a Query algorithm is decomposed into various parts depending upon the data required. Before passing query to algorithm, first check whether the current data is available in cache or not; if it is, then fetching data from the server. If some of data are available in local cache, then decompose query in such a way that data not available in cache should only be fetched from semantic cache [1]. To increase the power of semantic cache, we use the object relational model that made the query on complex structure as efficient [7]. ORDBMSs deliver the lowest access time for development and for greatest performance combination when using objects because they stored objects on disk and have the translucent program integration with object programming languages [5]. Performance is boosted by storing objects directly on disk which excludes impedance mismatch. Development period are reduced because there is no need to program the caching for the application programs and there is only one model to develop [10].

In this approach, we mainly focus on the object relational query matching on complex data and structures which have many tuples to increase the complexity of query matching and they are time consuming. In this research, we proposed the approach ORDBMSs in semantic cache that reduce the cost and use less time for result; we easily increase the trust of database user by using his model. Data latency

and workload can easily be distributed and handled. Rest of the chapter is organized as follows: Section 2 contains review of literature and related work in the field. Section 3 contains proposed work and results are obtained in Section 4, while Section 5 concludes the chapter.

2. Review of literature

In this section, a comprehensive survey of cache is presented. In Sections 2.1–2.3, the concept and work of semantic cache on query is discussed. Sections 2.4 and 2.5 are dedicated to databases, while the related work is given in Section 2.6.

2.1 Cache

It comprises of small-sized type of volatile memory like the memory of computer that is useful in terms of providing high speed data while having an easy access to the processor and storage to install programs, applications, and data that are frequently used by the computer. It is considered to have memory that is fastest and is placed on the motherboard directly connected to the processor or Random Access Memory. Cache that has pronunciation as "cash" neither "catch" nor "cashay," saves information that is recently used for it to have accessibility later. A PC memory with short access time utilized for capacity of every now and again or as of late utilized directions or information called likewise reserve memory [11]. PCs consolidate a few unique kinds of storing with a specific end goal to run more productively, in this manner enhancing execution. There are few of the caches that comprise of browser cache, disk cache, memory cache, and processor cache [13].

2.1.1 Browser cache

The webpage data by default are found in the browser cache. For instance, when the webpage is being visited, the browser might cache the HTML, images, and any CSS or JavaScript files that are being referred by the page. When the website is accessed by different pages and of utilization similar pictures, CSS, or JavaScript, your program will not need to redownload the records. Rather, the program can basically stack and store them on a local hard drive from the cache [4].

2.1.2 Memory cache

During the time of a running application, there is a chance that is cache of data in system memory or Random Access Memory. The example is if there is a video project you are working on, the video clips and audio tracks from the hard drive into Random Access Memory may get loaded on the video editor, since this can reduce the delay while importing the files and editing them and RAM has easier accessibility than hard drive [4].

2.1.3 Disk cache

The HDDs and SSDs present have a small amount of Random Access Memory that fulfills the need of disk cache. The typical disk cache of 1 TB has 32 megabytes, while a 2 TB hard drive may have a 64 MB cache. Therefore, the little measure of Random Access Memory can have a major effect in the execution of drive. The example is of when an envelope is opened with a substantial number of records, the

referring of documents might be naturally spared. The list of files is loaded instantly despite taking some time to appear when the folder is opened [11].

2.1.4 Processor cache

They are smaller in size as compared to disk cache. The reason is of the processor cache that has some tiny blocks of data that are basically instructions that are used frequently and can be accessed by the CPU quickly. Present day processors frequently contain a L1 reserve that is appropriate by the processor and a L2 store that is marginally further away. The L1 reserve is the littlest (around 64 KB), while the L2 store might associate with 2 MB in measure. Some top of the line processors even incorporate a L3 store that is bigger than the cache L2. The data might get moved to the level that is low to access it faster when the processor tries to access data from a level that is higher in caches [4]. The caching done in background will not get noticed. However, browser cache is the only cache that can be controlled. There is choice to view the settings of cache and change the size of browser and even empty it if there is a need [11].

2.2 Cache levels

Following are different cache levels and their details.

2.2.1 L1 cache

L1 (Level 1 cache) is a memory bank built into the CPU chip. Also known as the "primary cache," the cache that has the fastest memory is L1 in computer and is closer to processor.

2.2.2 L2 cache

L2 (Level 2 cache) has a memory bank that is made inside the CPU with a package present inside the component or is built on motherboard. The L2 cache feeds the L1 cache, which feeds the processor. The L1 memory is better than L2; basically L2 is slower than L1.

Figure 1 illustrates the working of cache on database—to access the data with the help of cache and improve the answer time of query.

2.3 Semantic cache

This works for the caches query result. And the elements that are present in the semantic cache are known as the regions or segments [14]. This term semantic caching is derived from the semantics of SQL queries that work for systematically handling the information of cache and building conclusions of the availability or unavailability of query results in the cache [12].

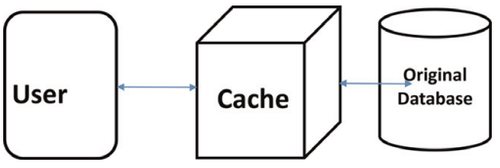

Figure 1.
Cache working.

The performance of the client-server systems is improved by the caching at local clients. The novel caching scheme being introduced is hence called the semantic caching. The transformation of the semantic storing can enhance the efficacy of XML inquiry that is prepared in the Web condition [9]. Semantic storing increases reserved information with a semantic depiction of the information.

These semantic depictions can be utilized to enhance execution time for comparable inquiries by recovering little information from reserve and issuing a leftover portion question for the rest. Benefits of semantic reserving include low network overhead, independence of physical format of the database, decreased system activity, and the capacity to answer a few inquiries without reaching the server. For workloads that are less complex, there is a need to maintain efficacy of the query processing by cautious coding of queries that are remainder at the server. For workloads that are very complex, using very complex workloads, there is a display of semantic caching that works better in a variety of applications specifically in the environments that were constrained [9].

2.3.1 Semantic cache scenarios

Semantic cache answers the query in different scenario's which are described below [15] in **Figures 2–5**, respectively. The scenarios are, namely

- full answer;
- partial answer; and
- no answer.

Example:

The semantic data are being extracted from the query while there is addition to this semantic data that will be used for more matching and the cache [16]. This high-power semantic fragment reserve is versatile, which means that, as and when the client is entering the inquiry for which the appropriate response is to be discovered, the applicable characteristics of the database will be populated in the store. The part of the cache that is semantic basically the highlights and the content which is refined just add quality in boosting the performance in a manner that is convincing and exuberant [14].

In case of the semantic cache, the semantics stored on the cache are compared to the input user query and subject to availability of data, the decision is taken. It is carried out by two processes known as splitting that involves division of query based on its clauses and rejecting if a certain clause is missing, e.g., WHERE

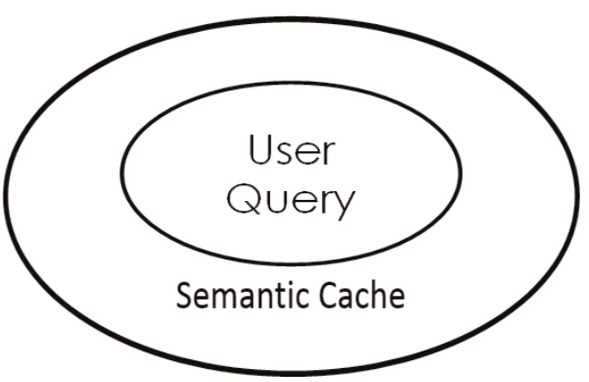

Figure 2.
Semantic cache full answer.

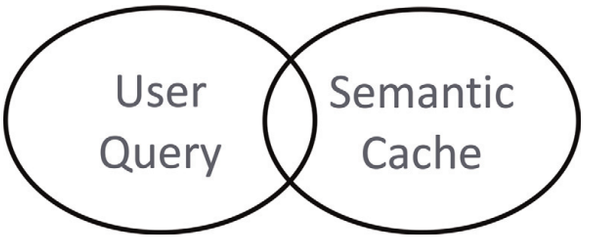

Figure 3.
Semantic cache partial answer.

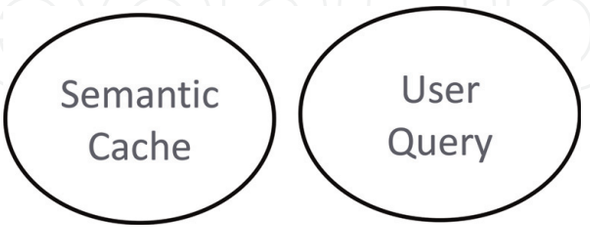

Figure 4.
Semantic cache no answer.

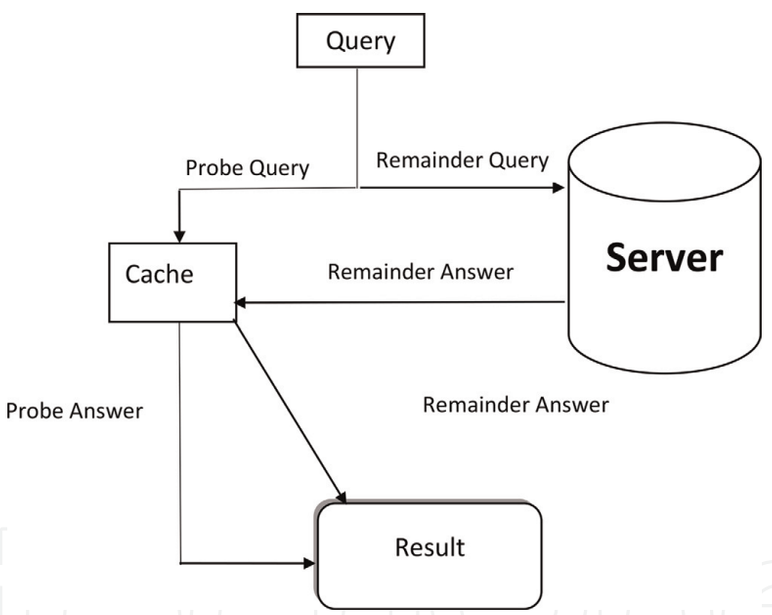

Figure 5.
Semantic cache working.

and so on [1]. The queries that are matched (overlapped) either fully or partially are answered locally by the semantic cache. Query processing and cache management are the main critical aspects of semantic cache, yet it performs way better than simple data (page, tuple) cache. Semantic caching provides the significance workload reduction in distributed systems, especially in mobile computing as well as improves the performance. However, the performance is purely based on the efficiency of its subprocesses like query trimming, indexing, etc. [1].

2.4 Relational database

A relational database is a category of database. It uses an arrangement that lets us to recognize and access data in relation to additional part of data in the database. Often, data in a relational database are organized into tables [5]. A relational

database management system (RDBMS) is a program that allows you to create, update, and administer a relational database [17]. Most relational database management systems use the SQL language to access the database. In RDBMS, the data are stored in the form of relations (tables) in a row-column architecture. It is comprised of records (rows) that are uniquely identified by a key attribute. There are several ways to access the stored data without manipulating the database relations as such [5].

Example:

In this example, a case study is used to understand the relational database and query is conducted on data model to understand working (**Tables 1** and **2**).

A query is conducted "Query: - Select Account=6 From Main account, Employee table" and for answer of query, every record is checked which is time consuming.

2.5 Object relational database

Object relational query processing is needed to speed up queries over object relational databases. We are here to define a couple of features mentioned in to characterize an ORDBMS. These structures are desired to model real-world problems in a method that is instinctive and easy for the developer and proposals noble performance for the application (**Figure 6**).

In this example, the query is answered directly by object which saves the time, and query efficiency is increased.

2.6 Related work

In [9], authors proposed an XML-based system "XPERANTO" for data representation and the access is duly retrieved from a native database for

Account no	First name	Last name	Amount
1	John	Doe	277$
2	Clay	Russell	586$
3	Albert	Luke	321$
4	Christina	Jorge	448$
5	Tim	Joe	520$
6	Dany	Clark	459$

Table 1.
Main account.

Account no	Emp-ID	Title	Branch
1	BW-123	Flipper	California
2	CA-448	Cashier	L.A.
3	DG-456	Manager	Washington
4	FA-114	Washer	London
5	DC-587	Doctor	Canada
6	HG-894	Plumber	England

Table 2.
Employee account.

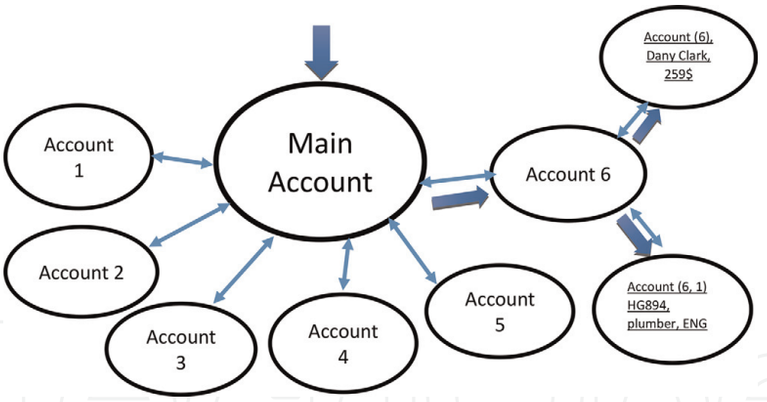

Figure 6.
ORDBMS working.

better accessibility. The system works as a middleware between XML and native database.

In [18], authors proposed a digital library and archiving system for educational institutes. The system takes advantage of ORDMS concept and builds a top layer XML object. These objects are kept in a library that can be accessed by client side QueryX engine duly executed by IBM domino server.

In [19], authors proposed a query optimization technique for RDF data stored in triplet format. The main idea was optimization of the SPARQL query based on the storage type, that is, adjacency list or matrix. It was concluded that the performance depends on the nature of data whether it is dense or sparse.

In [20], authors investigated the TYPE constraint for sake of query optimization in the context of frequent pattern mining. The idea behind this research was the data type that plays an important role in semantic association that increases the likelihood of its access.

Brown [10] presented the ORDBMS technique and investigated its properties related to flexible data access, functional improvement, enhanced efficiency, and organizational integration.

Author in [5] presents the object relational mapping (ORM) approach. The ORM refers to better data and transaction handling on a database using an object oriented approach. The investigation was conducted on a Java-based open source system "Hibernate," which is currently added to Microsoft model for .Net Systems.

In September 2007 [21], the Object Database Technology Working Group of the Object Management Group (OMG) issued a white paper that introduced the concept of an "object calculus" for ODBMSs that is analogous to "relational calculus" in RDBMSs.

In [22], the authors proposed the research of cache moves around in the scalability of new data-intensive environments and applications, and the trade-offs that are highly determined by the characteristics of these applications. Early work on information storing, for instance, concentrated on protest situated database frameworks supporting applications, for example, CAD/CAM; these frameworks had the coupling between the customers and server which took into consideration sharing of individual tuples or entire plate pages. The procedures utilized in examinations have been named physical reserving strategies.

In [23], the authors present the query-based services that do not entirely give out the physical layout of database; furthermore, customers have no power over the internals or interfaces; even application servers have just the data in the inquiries. Regarding the reserving models, the administrations should consequently be dealt with as self-sufficient inheritance frameworks even though they may dwell in best in class business database frameworks. In this condition, physical reserving

strategies are basically no longer pertinent as there is assumption coupling between client and server.

In [24], authors present the Object Relational Query Processing approach for optimizing the queries over ORDBMS. The approach was originally inspired by the object oriented paradigm.

In [25], authors present the idea of a three-level caching for efficient query processing in large Web search engines where a huge number of interactive data queries are posed in small fraction of time. Due to the volume of data access, semantic caching was a plus in efficiently handling data for sake of improving response time and reducing Web traffic. To keep up with this immense workload, large search engines employ clusters of hundreds or thousands of machines, and several techniques such as caching, index compression, and index and query pruning are used to improve scalability. Each level equips the higher level for better accessibility and locality [26].

3. Research methodology

In this section, the proposed model of the work is explained in Section 3.1. Then notation table that is used to understand the model in Section 3.2 and algorithm on query matching in Section 3.3 are discussed, and then a case study is conducted in Section 3.4.

3.1 Proposed model

Figure 7 is used to describe the proposed model, that is, the object relational query as example. Suppose we have a query Select Selection department, Section Marks, Grade From Enrollment Where Student S.Name='Clay' And Section Department='CSCI'.

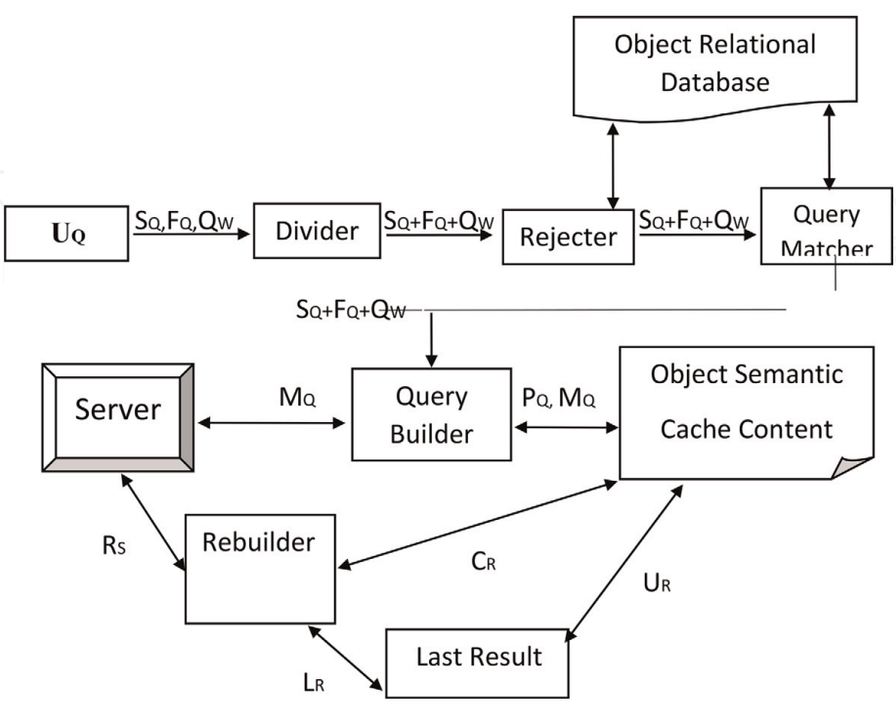

Figure 7.
Proposed model.

Notation	Details
SC	Cache segment
UQ	User query
SQ	***Select*** part of query
FQ	***From*** part of query
WQ	***Where*** part of query
MQ	Modify query
RQ	Remainder query
PQ	Probe query
PA	Predicate attribute
AS	Attribute of segment
PS	Predicate of segment
RS	Relation of segment
CS	Content of segment
AK	Key attribute of segment
SA	Same attributes
RS	Result from server
DA	Difference attribute
LR	Last result
CR	Server result
QR	Query result

Table 3.
Notation table.

All the notations used are enlisted in **Table 3**.

3.2 Proposed algorithms

Algorithm 1:-	The pseudo code of proposed algorithm to match the query
Purpose:-	To enhance the query matching approach on the ORDBMS over semantic cache
Input:-	User query, Semantic cache
Output:-	Result of User Query (Probe Query and Remainder Query)
PROCESSING:-	Get the query from user and go to query splitter
STEP 1:-	DIVIDE _QUERY (**UQ**)
STEP 2:-	Rejecter: - CHECK _REJECTIONS(**SQ**+ **FQ**+ **PA**)
STEP 3:-	IF (Reject= False) I. **SA**, **DA**:= MATCH _ SELECT_CLAUSE (**SQ**) a. IF (**DA** != Empty) b. **RQ**1 = π **DA**σ**PQ** (QR) II. Else rq1= Null

	III. IF (**SA** != Empty) a. IF (! (QPA ⊆ **SP**)) **MQ** = GEN_AMEND_QUERY() b. ELSE **MQ** = Null IV. IF (QP ==>SP) a. PQ:=π**S**AσPS (CS) b. RQ2: Null V. ELSE (QP ^ SP) is Satisfiable PQ : πSAσPS (CS) VI. ELSE IF (QP ^ SP) is Unsatisfiable Pq: Null Rq2 = πCA σSP (QR) a. ELSE b. PQ:= Null c. RQ2:= Null d. LR: = PQ + RQ + RQ2
STEP 4:	ELSE Query Is Incorrect

Algorithm 2	**DIVIDE _UQ()**
Input	**UQ** (Query from user)
Output	**SQ, WQ, FA**
Procedure	**SQ**: - SELECT CLAUSE **WQ**:- WHERE CLAUSE **FA:**- FROM CLAUSE Return:- SQ, WQ, FA

Algorithm 3	CHECK _REJECTIONS(**SQ**, **FQ**, **PA**)
Input	**UQ** (User Query)
Output	**SQ, FQ, PA**
Procedure	I. If all attributes of **SQ** present in schema II. If relation of **FQ** present in schema III. 5 IV. If **PA** is present in schema Return false Else return true V. Else return true VI. Else return true

3.3 Case study

Following schema is taken as a case study to demonstrate the proposed approach. In this regard, following object relational database query is posed. UQ2: "Select (selection) Section, Department, Marks, Grade From Enrollment Where Student S.Name='Clay' And Section Department='CSCI'".

UNIVERSITY							
STUDENT				**ENROLLMENT**			
S.Name	S.ID	Age	Gender	Section	Department	Marks	Grade

Enrollment		**Student**	
S. no	**Cache segment**	**S. no**	**Cache segment**
C1	S.ID	C14	Section
C2	S.name	C15	Department
C3	Gender	C16	Marks
C4	Age	C17	Grade
C5	S.ID, S.name	C18	Section, department
C6	S.ID, gender	C19	Section, marks
C7	S.ID, age	C20	Section, grade
C8	S.ID, S.name, gender	C21	Section, department, marks
C9	S.ID, S.name, age	C22	Section, department, grade
C10	S.ID, gender, age	C23	Department, marks
C11	S.name, gender	C24	Department, grade
C12	S.name, age	C25	Marks, grade
C13	Gender, age	C26	Grade, marks

Table 4.
Cache segments on relation.

For the above given case study of university, there are 26 possible cache segments of the enrollment and student relation. In other words, we can say that 13 are made against the enrollment relation and 13 for the student as in given **Table 4** according to given formula $2^n - 1$ [1].

4. Results and discussion

The discussion on the case study in Section 4.1 and the comparison on the case study in Section 4.2 are conducted.

4.1 Discussion

In the example, there are 30 possible enquiries that make separate segments. But in ORDBMS, the reference is used toward accessing the query result and the reference is added on the row [7]. The given two object oriented user query on possible segments are as follows:

UQ1:- ***Select*** Section department, Section Grade, **F*rom*** Student **W*here*** Student S. Name='Bursch' And Student Age='21'.

UQ2:- ***Select*** Selection department, Section Marks, Grade **F*rom*** Enrollment **W*here*** Student S. Name='Clay' And Section Department='CSCI'

As from above Object queries, UQ1 is rejected as initial state from query rejecter SQ is not coordinated with attributes of Student relation. Now let us assume from UQ2 over projected architecture as with respect to cache segment of Object relational query from **Table 4**. Query split function splits the query into segment with reference as below:

SQ: -{Selection department, Section Marks, Grade}
WQ: -{Enrollment, Student}
FA :-{Student S. Name='Clay' and Section Department='CSCI'}

Rejecter receives these three {SQ, WQ, FQ} and passes it to decider after checking the validity. Then, the decider checks the availability of required attributes by applying the proposed approach.

Here, for simplification, we assume that there exist two segments S12 and S16, for enrollment and student. So, the SA and DA will be composed as follows:

SA = {Department,Marks}

DA = {Grade}

After combining difference and common attribute, query will generate and send the following remainder query to the server.

RQ= Select Grade From Enrollment Where Section Department='CSCI

Common attributes with WQ and FQ will be sent to LQG, whereas probe and remainder queries will be produced by LQG based on similarity with segment on cache [14]. Note that here SQ will be equal to SA. So, probe and remainder queries are given below:

PQ= Select Selection department, Section Marks, From (CS)

RQ= Select Grade from Enrollment Where Section Department='CSCI.

Here, modify query (MQ) is null because PS ⊆ SQ. This process of takeout probe and remainder query will be continued with entirely of segments that are visited or remainder queries become null. Query generator sends all the probe queries to the cache content and final remainder query to the server to retrieve data.

As a final point, rebuilder obtains CR from cache and RS joined to build LR and the semantics in the cache will be updated accordingly.

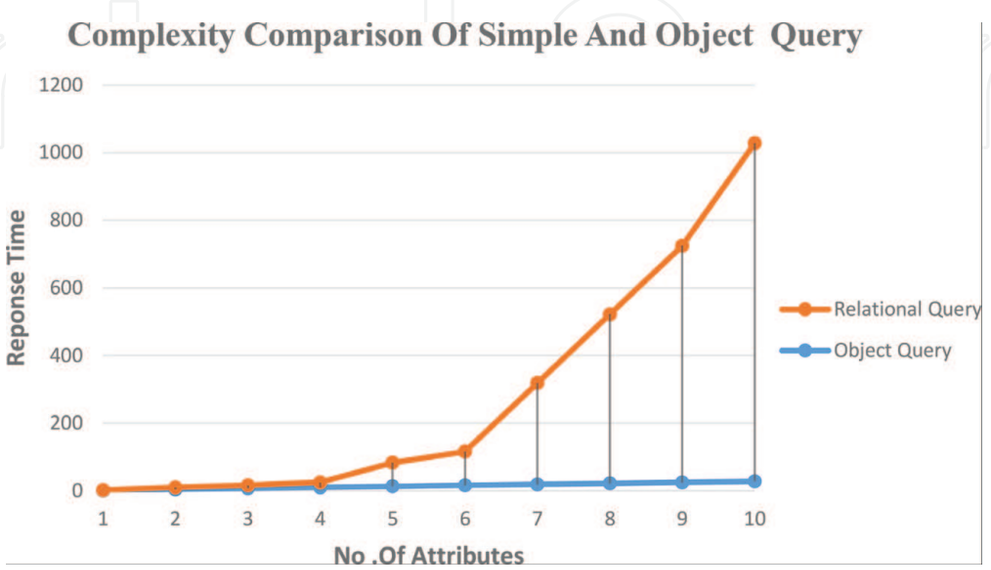

Figure 8.
Complexity comparison.

4.2 Complexity comparison

This section provides comparison between previous work on RDBMS with semantic cache and proposed query matching scheme on ORDBMS with semantic cache [7]. We have used workload parameter, such as response time, no of attribute more detail is given with the help of **Figure 8** present the response time through number of attributes. Comeback time is purely calculated on the bases of complexity expression use as previous (n vs. 2^n-1).

In **Figure 8**, the comparison result is displayed, which is used to show difference between relational query on relational data model with semantic cache and object relational database query with sematic cache, and response time is getting better on object relational query and object relational database can have ability to answer the complex data type. The retrieve time of query on ORDBMS is better and efficient; the query matching approach has improved the working procedure of the object relational query.

5. Conclusion

In this proposal, we talked about the significance of ORDBMS query for associations and organizations. We featured our approaches for outlining model and approach algorithm. Additionally, we talked about contextual analyses of executing RDBMS in online store situations and their methodologies of the implementation.

This exploration, in current stage, centers on outlining and available information which will help for the most part in basic leadership process that is identified with the advertising. We found that the greater part of works in this field have been given diverse ways to deal with the choice of perspectives to appear considering query upkeep cost and time consuming.

The investigations demonstrate that the proposed model can be incorporated with the existing models since it limits the arrangement of perspectives before appearance process.

In this research, we proposed an efficient scheme to reduce the query execution cost by making the query matching process swift. Moreover, in this era, every organization required the records in short time in the presence of big data, data lake, Teradata, etc. On the other hand, the organizations do not want to change their current systems due to the reasons like data losses, delays, and other cost-related issues. To avoid these issues, proposed advanced level query matcher can be a good alternate.

To fix this issue, we present a technique for Query Matcher and semantic cache process over object relational database. We use object query on relational database.

Now we provide solution for decision-makers or user of traditional database that can enhance speed and cost and optimize query matching. We can manage large queries on data set with matching approach and save these results dynamically in semantic cache which updated on run time. Rather accessing the whole large data set from object relational database we pull data from Semantic cache where similar queries answer before that reduce time and system delay. This will increase the confidence of database users.

Author details

Hafiz Muhammad Faisal[1], Muhammad Ali Tariq[1], Atta-ur-Rahman[2*], Anas Alghamdi[2] and Nawaf Alowain[2]

1 Barani Institute of Information Technology, PMAS Arid Agriculture University, Rawalpindi, Pakistan

2 Department of Computer Science, CCSIT, Imam Abdulrahman bin Faisal University, Dammam, Saudi Arabia

*Address all correspondence to: aaurrahman@iau.edu.sa

References

[1] Ahmad M, Qadir MA, Sanaullah M, Bashir MF. An efficient query matching algorithm for relational data semantic cache. In: 2009 2nd International Conference on Computer, Control and Communication; Karachi. 2009. pp. 1-6

[2] Ahmad M, Qadir MA, Sanaullah M. Query processing over relational databases with semantic cache: A survey. In: 2008 IEEE International Multitopic Conference, Karachi. 2008. pp. 558-564

[3] Fagin R. Multivalued dependencies and a new normal form for relational databases. ACM Transactions on Database Systems. 1977;**2**(3):262-278

[4] Trappey AJC, Lee C, Chen W, Trappey CV. A framework of customer complaint handling system. In: 2010 7th International Conference on Service Systems and Service Management; Tokyo. 2010. pp. 1-6

[5] O'Neil EJ. Object/relational mapping 2008: Hibernate and the entity data model (edm). In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08). New York, NY, USA: ACM. pp. 1351-1356

[6] Jeudy B, Rioult F. Database transposition for constrained (closed) pattern mining. In: Goethals B, Siebes A, editors. Knowledge Discovery in Inductive Databases. KDID 2004. Lecture Notes in Computer Science. Vol. 3377. Berlin, Heidelberg: Springer; 2005

[7] Risch T. Introduction to object-oriented and object-relational database systems. In: Uppsala Data Base the Laboratory. 2016

[8] Subramanian M, Krishnamurthy V. Performance challenges in object-relational DBMSs. In: Oracle Corporation Redwood Shores CA 94065. 1999

[9] Carey M, Kiernan J, Shanmugasundaram J, Shekita E, Subramanian S. A Middleware for Publishing Object-Relational Data as XML Documents. 2003

[10] Brown PG. Object-Relational Database Development: A Plumber's Guide with Cd rom. Upper Saddle River NJ, USA: Prentice Hall PTR; 2000. ISBN 0130194603

[11] Coronel C, Morris S. Database Systems: Design, Implementation & Management. San Francisco, CA: Cengage Learning; 2016

[12] Ahmad M, Qadir MA, Ali T, Abbas MA, Afzal MT. Semantic cache system. In: Semantics in Action-Applications and Scenarios. Rijeka: IntechOpen; 2012

[13] Thakur D. What is cache memory | types of cache memory. In: Computer Notes Blog

[14] Chidlovskii B, Roncancio C, Schneider ML. Semantic cache mechanism for heterogeneous web querying. Computer Networks. 1999;**31** (11–16):1347-1360

[15] Ahmad M, Qadir MA, Ali T. Indexing for semantic cache to reduce query matching complexity. Journal of the National Science Foundation of Sri Lanka. 2017;**45**:13. DOI: 10.4038/jnsfsr.v45i1.8033

[16] Godfrey P, Graz J. Answering queries by semantic caches. In: International Conference on Database and Expert Systems Applications. Berlin, Heidelberg: Springer; 1999. pp. 485-498

[17] Luo et al. The Query-Based Services Give Out the Physical Layout of Database. 2002

[18] Atta-ur-Rahman, Alhaidari FA. The digital library and the archiving system for educational institutes. Pakistan Journal of Information Management and Libraries (PJIM&L). 2019;**20**(1):94-117

[19] Atta-ur-Rahman FAA. Querying RDF Data. Journal of Theoretical and Applied Information Technology. 2018; **26**(22):7599-7614

[20] Ahmad M, Farooq U, Atta-ur-Rahman AA, Dash S, Luhach AK. Investigating TYPE constraint for frequent pattern mining. Journal of Discrete Mathematical Sciences and Cryptography. 2019;**22**(4):605-626

[21] Object Management Group (OMG). The Object Database Technology Working Group Present, "Object Calculus" for ODBMSs. 2007

[22] Dewitt et al. (1990), Carey et al. (1991) and Franklin et al. (1996) proposed the research of cache moves around in the scalability new data-intensive environments

[23] Jaluta I, Bazina N. Cache consistency in adaptive page-server database systems. 2014. pp. 1-5. DOI: 10.1109/GSCIT.2014.6970098

[24] Liu B, Lee WC, Lee DL. Distributed caching of multi-dimensional data in mobile environments. In: Proceedings of the 6th International Conference on Mobile Data Management. ACM; 2005, May. pp. 229-233

[25] Long X, Suel T. Three-level caching for efficient query processing in large web search engines. In: CIS Department Polytechnic University Brooklyn, NY 11201

[26] Leung CKS, Brajczuk DA. Efficient algorithms for mining constrained frequent patterns from uncertain data. In: Proceedings of the 1st ACM. 2009