Received December 29, 2021, accepted January 24, 2022, date of publication January 31, 2022, date of current version February 7, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3148131

Optimizing the Performance of Data Warehouse by Query Cache Mechanism

CH ANWAR UL HASSAN[1], (Member, IEEE), MUHAMMAD HAMMAD[1], MUEEN UDDIN[2], (Senior Member, IEEE), JAWAID IQBAL[1], JAWAD SAHI[3], SADDAM HUSSAIN[2], AND SYED SAJID ULLAH[4]

[1]Department of Computer Science, Capital University of Science and Technology, Islamabad 44000, Pakistan
[2]School of Digital Science, Universiti Brunei Darussalam, Gadong, Bandar Seri Begawan BE1410, Brunei
[3]Department of Computer Science, COMSATS University, Islamabad 44000, Pakistan
[4]Department of Electrical and Computer Engineering, Villanova University, Villanova, PA 19085, USA
[5]Department of Information and Communication Technology, University of Agder (UiA), 4898 Grimstad, Norway

Corresponding authors: Saddam Hussain (saddamicup1993@gmail.com) and Syed Sajid Ullah (sullah1@villanova.edu)

ABSTRACT Fast access of data from Data Warehouse (DW) is a need for today's Business Intelligence (BI). In the era of Big Data, the cache is regarded as one of the most effective techniques to improve the performance of accessing data. DW has been widely used by several organizations to manage data and use it for Decision Support System (DSS). Many methods have been used to optimize the performance of fetching data from DW. Query cache method is one of those methods that play an effective role in optimization. The proposed work is based on a cache-based mechanism that helps DW in two aspects: the first one is to reduce the execution time by directly accessing records from cache memory, and the second is to save cache memory space by eliminating non-frequent data. Our target is to fill the cache memory with the most used data. To achieve this goal aging-based Least Frequently Used (LFU) algorithm is used by considering the size and frequency of data simultaneously. The priority and expiry age of the data in the cache memory is managed by dealing with both the size and frequency of data. LFU sets priorities and counts the age of data placed in cache memory. The entry with the lowest age count and priority is eliminated first from the cache block. Ultimately, the proposed cache mechanism efficiently utilized cache memory and fills a large performance gap between the main DW and the business user query.

INDEX TERMS Data warehouse, optimization, big data, query optimization.

I. INTRODUCTION

Nowadays, businesses are growing rapidly, and the influence of Data Warehouses (DWs) has been increasing day by day, being used by several organizations. High customer demands are driving DWs to make fast business decisions based on the latest information [1]. Dealing with a large amount of data can slow down the performance, especially in data fetching, which will affect the user experience. Several methods have been proposed to optimize data execution and data processing. In [1], the author used query cache algorithms for optimizing the process of data execution.

The amount of data is increasing day by day due to the usage of software technology. By using technology, businesses are automated and things are being recorded to make it easier and for fast and secure retrieval. To, later on, analyze this huge volume of historical data, identify the hidden patterns and make successful business decisions. As we know, the amount of data is growing day by day, so due to the large volume of the data, the data retrieval rate is also affected. To optimize the performance of accessing data, practitioners apply different approaches to make it easier and faster to access. In this regard, researchers are focused on the most advanced query acceleration and cache memory techniques. Researchers used the cache-based method to enhance the data access performance of the DW. The cache method can maintain the performance gap by considering the execution time between the main database and the cache memory. In order to efficiently utilize cache space, various techniques are used. Least Frequently Used (LFU) and Least Recently Used (LRU) are the two most commonly used algorithms for cache replacement. In the last few decades, a variety of combinations of LRU and LFU have been proposed, and these combinations are used to come up with the optimal solutions

The associate editor coordinating the review of this manuscript and approving it for publication was Xueqin Jiang.

for cache replacement [2]. It is very hard to add all objects into cache memory in order to make it efficient [3]. An extensive study on cache management has led to a large variety of algorithms, including both general-purpose and domain-specific ones. To maximize the utilization of cache memory, a competent algorithm should always keep the most popular blocks in cache [4]. The most commonly used algorithms for cache replacement are: Least Frequently Used (LFU), Most Frequently Used (MFU), and Least Recently Used (LRU) [5]. Cache memory is placed with the Online Analytical Process (OLAP) for answering client-based queries, and the results of business users are fetched from the OLAP server and stored in cache memory.

Several solutions for DW query cache optimization were proposed by the researchers. However, these approaches have some limitations, such as data skew exploiting, poor load balancing resulting in a long response time, ad hoc query constraints, query repositioning, textual, spatial, and temporal data characteristics, and parallel execution of complex queries, which increases computational resource usage and affects execution time. Along with this, some authors proposed hardware-based solutions such as cloud-based environments and modern processors for optimization of DW query processing. To overcome these limitations, in this paper, we proposed the query cache mechanism for optimizing DW performance. Following are some of the major contributions of your research.

- We designed a cache mechanism intending to build an efficient cache replacement technique.
- We used the LFU algorithm for query caching while considering the size and frequency of the data at the same time. The ability of the cache method to maintain a performance gap by considering the execution time between the main database and cache memory.
- The proposed method is cost-effective and doesn't use any hardware like modern processors or cloud-based processing environments for query optimization. We also can't reposition and concatenate the queries to avoid the usage of high computational resources.
- The proposed method has reduced the analytical query response time and data entry interference caused by the simultaneous and long-term execution of queries.

By applying the Query cache mechanism, we save the frequently executed data with data size. Based on frequency and size, the most frequently used tables will be given high priority, and data with low priority will be trashed from cache memory. In this way, cache memory will be saved from non-frequent data and improve the analytical query response by responding to user queries efficiently, dealing with limited and most-demanded data.

The rest of the paper is structured as follows: Section II presents related work. The LFU replacement strategy and size-based replacement strategy are discussed in Section III. The proposed cache mechanism is described in Section IV. The DW architecture is given in section V, and the algorithm and pseudo code of the proposed system is presented in section VI. Finally, section VII concludes the paper.

II. RELATED WORK

Nowadays, fast retrieval of information from the DW is the need of businesses to take timely decisions. In this regard, many researchers have proposed different solutions to effectively fetch data from OLAP servers to meet the needs of BI users. In computing, a DW works as an enterprise DW to analyze the huge historical data and extract useful information from it to make a successful business decision [6]. Day by day, with the advancement of technology, practitioners contribute to making the DW more efficient and effective by enhancing the efficiency of OLAP using algebraic expressions. In [7], [8], authors optimize the performance of OLAP by proposing the data cube model and algebraic expression to support its operations and concisely express and evaluate complex OLAP queries against the distributed DWs, respectively.

The amount of data is increasing day by day, with the increasing amount of data, it is difficult to extract valuable information or to gain insight from the data for effective business decisions. In this regard, different strategies were proposed by practitioners to gain insight from the data and retrieve the information quickly by optimizing and enhancing the ETL process and query processing [9]. Query processing techniques are commonly used in DWs for the fast retrieval of information. Researchers achieve query processing by running parallel queries [10] or by employing group-by or having-based conditional queries [11] efficiently to improve querying efficiency and performance on large amounts of data [12]. In [13], the authors show the query cache method used before the ETL process to increase the performance of DW. In this method, data is stored in cache memory after fetching user query results from the DW. If new data is requested to be fetched from the DW, it first checks it from the query cache memory to reduce the response time. SQL query acceleration is presented in [14]. Using the FPGA-based approach, they speed up database queries using joins significantly. This work is focused on and achieved by optimizing hardware-based filtering.

In [15], [16], the authors optimize ETL process modeling by introducing a control-flow-based approach, structuring ETL processes in its application to a business case that starts from business fact identification, and using the combination of parallelization and shared cache memory to optimize the ETL performance of DW respectively. Average Least Frequency Used Removal (ALFUR) is employed and contrasted with other cache replacement techniques in [17]. The ALFUR technique outperformed as compared to LFU and LRU. Results show that ALFUR does not perform well as compared to LFU in a case where the hitting ratio cache size increases. However, the size of the data with cache replacement is not considered. If we have a large amount of data that has been fetched and cached into query cache memory, in this case, these methods cannot perform effectively, and the

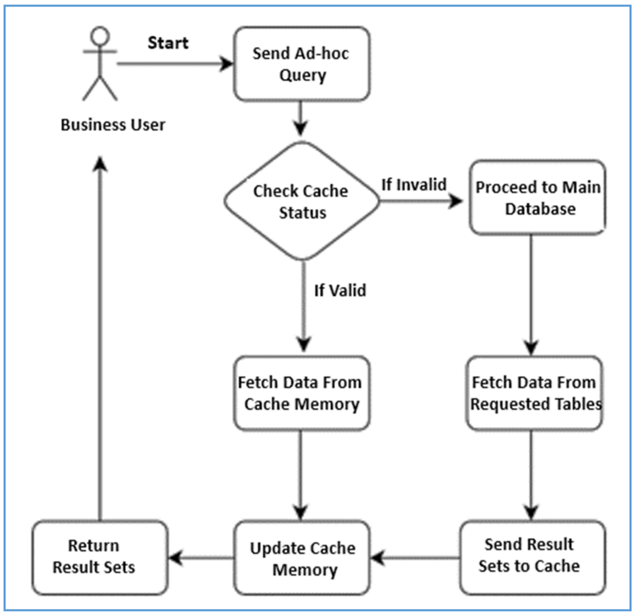

FIGURE 1. Flow chart of cache mechanism.

purpose of the query cache is evaded to use it for fast access. This slows down the performance of cache methods [18]. Cache techniques vary from task to task. It depends upon the different parameters to choose a suitable cache algorithm for a specific domain.

In [19], a cache-based mechanism is implemented using Spatial Data Stream Warehouse (STAR) to speed up query processing by combining object-based and query-based caching techniques. STAR, on the other hand, supports queries made up of temporal data features, ad-hoc query constraints, and aggregate functions. The authors of [20], [21] transform the set of queries into a new one by satisfying memory constraints to avoid unnecessary re-computation in the distributed environment and use the cloud environment for distributed queries to optimize query performance through data processing time, respectively. A hardware-based approach is proposed to manage the growing workload with modern processors to optimize the cache performance [22].

The author improved the data quality by using the query cache to enhance the performance of ETL processing by filtering the dirty, unclean data. However, the author only checks the state of the query. If it is saved, then it is valid. Otherwise, the query needs to be processed again, which will increase the processing time [23]. In addition, analyze cache behavior and implement database operators that are efficient in the presence of skew. Nevertheless, the author tries to solve the data skew problem by repositioning the cache data that affects the parallel execution of complex database queries, which leads to poor load balancing and high response times [24].

III. PROPOSED LFU REPLACEMENT ALGORITHM

The fundamental characteristic of the LFU method is that it keeps monitoring the number of times a block is referenced in memory. When the cache becomes full and there is no more space for a new object, the system will expel the object from the cache having the lowest reference frequency and create space for new items. For LFU, data is inserted into the cache as shown in Figure 1.

The LFU experiences the problem of cache pollution. The object with the extensive reference accounts is replaced even if the cached objects are not re-accessed again. [25] By adding an aging faction given by work [size-based cache], we can overcome this cache pollution and save memory from non-frequently used data. Unlike LFU, implemented aging-based LFU adds key value to cached data based on its age. LFU calculates the key-value of object g using Eq. 1.

$$k\ (h) = A + F(h) \tag{1}$$

where F(h) is the frequency of the requests of data (h), while A is a dynamic aging factor. *A* is initialized to zero and then updated to the key value of the last removed object. The main disadvantage of the LFU replacement algorithm is that some data should be in cache even without using them again for a long time [25].

A. SIZE-BASED CACHE REPLACEMENT

The size-based cache replacement policy is one of the most commonly used approaches for caching. When space for a new object is needed, it replaces the new data with the one having the largest size. When cache space is utilized and new data for storage requires a cache, the data having the lowest key value is eliminated [26]. When the user requests the data g, the Greedy-Dual-Size (GDS) algorithm suggested by Cao and Irani assigns the key-value of data g as shown in Eq.2.

$$k\ (h) = A + \frac{\mathrm{C(h)}}{\mathrm{S(h)}} \tag{2}$$

where C(*h*) is the cost of retrieving data (*h*) from the server into the cache; S(*h*) is the size of the date (h), and A is an aging factor. *A* in GDS starts at zero and is updated to the key value of the last replaced object. The key value *k* (*h*) of data (*h*) is updated using the new A value since data (h) is accessed again. Thus, the other equation variable definitions are shown in Table 1.

TABLE 1. Equation variables definition.

Variables	Definition
g	Requested data
K(g)	Key-value of data (g)
A	Age of the data
F(g)	Frequency of the data (g)
C(g)	Execution time
S(g)	Size of data (g)

B. CONSIDERING SIZE, FREQUENCY, AND AGING

Data frequency is the number of times the data value occurs, and data aging is said to be the process of removing the old data from the storage in order to reuse the space for future backups. By considering the frequency, size, and age of the data, the following Eq. 3 applies.

$$k(h) = A + F(h) * \frac{\mathrm{C(h)}}{\mathrm{S(h)}} \tag{3}$$

becomes where the age of data A is multiplied with frequency F(h) before cost and size calculation.

IV. PROPOSED CACHE MECHANISM

We have proposed a query cache mechanism that will retain all records of the queries that are requested once from the OLAP server by the business users. This method will keep setting the priorities of the results fetched from the OLAP database. Most demanded records have high priorities. This process is done by using aging-based LFU while considering the size of the data. Moreover, this cache mechanism holds two states: a valid and an invalid one. When a query is requested by the business user, it first checks the state of whether it is valid or invalid rather than directly hitting the cache memory. If the requested data is already present in cache memory, then the state is valid and the cache method allows the query to move forward and proceed. Ultimately, this will save the execution time of cache memory.

A. STATES OF THE CACHE MECHANISM

1) INVALID

The state of the cache is invalid when data is not present in the cache memory. There are two reasons for data not being available in the cache memory: One is when data is not fetched from the main database even once, and the second is when data is fetched once and still states that it is invalid, which means data is eliminated from cache memory for not being accessed frequently enough to save cache space.

2) VALID

The cache state is valid when demanded data is not eliminated and available in the cache memory. The results will be fetched from the cache memory and sent to the business users.

B. PROPOSED ALGORITHM

We have performed the simulation of the proposed algorithm using the CORE i5-8265U @ CPU-1.6GHz (8 CPUs) 1.8GHz system, 8192 MB of RAM, and SSD. We configured the ETL-based DW and then obtained the data from the created DW using the OLAP servers. When a user enters the query for the first time, it takes time for execution, as the query is executed on amps. The most frequent data is placed in the cache memory to reduce the execution time next time by directly accessing records from the cache memory, and the second is to save cache memory space by eliminating non-frequent data. Our proposed algorithm considered the size, frequency, and age of the data using the LFU approach.

Algorithm 1 Algorithm Pseudo Code

```
Request new data (NewData);
while Check cache status do
    if Cache status = Valid then
        Proceed towards cache memory;
        Fetch requested data;
        Store C(g) Fetching cost/ Execution time of data;
        Store S(g) Size of data;
        Update cache memory K(g);
        Update variable values A(g) ++ AND
        Increment Variable F(g) ++ K(g) = A(g)+ F(g)
        * C(g)/S(g);
        Return Result Sets
    else
        Cache status = Invalid;
        Proceed towards main database;
        Fetch requested tables;
        Send result sets to cache memory;
        Update cache memory K(g)
        Update variable values A(g) ++ AND
        Increment Variable F(g) ++ K(g) = A(g)+ F(g)
        * C(g)/S(g);
        Return Result Sets
    end
end
```

C. DATA WAREHOUSE ARCHITECTURE WITH PROPOSED CACHE MECHANISM

In the proposed solution of cache replacement technique, cache mechanism is placed with OLAP after ETL operations. When Ad-hoc queries from business users are requested, it first checks the result in cache memory as shown in Figure 2. An ETL framework of DW comprises of three back-to-back utilitarian steps: extraction, transformation, and loading which is described as follows;

- Extraction.
- Transformation.
- Loading.

1) EXTRACTION

The extraction of data from source systems like Customer Relationship Management (CRM), Enterprise Relationship Management (ERP), and other Online Transaction Processes (OLTP) is the collection of all the data into a single integration point. This is the first step of DW. After collecting all the data from the source systems, the data is saved into a single database. Because data is collected from different source systems, which may not be collaborative before the DW environment, So, there is a need to clean data errors to

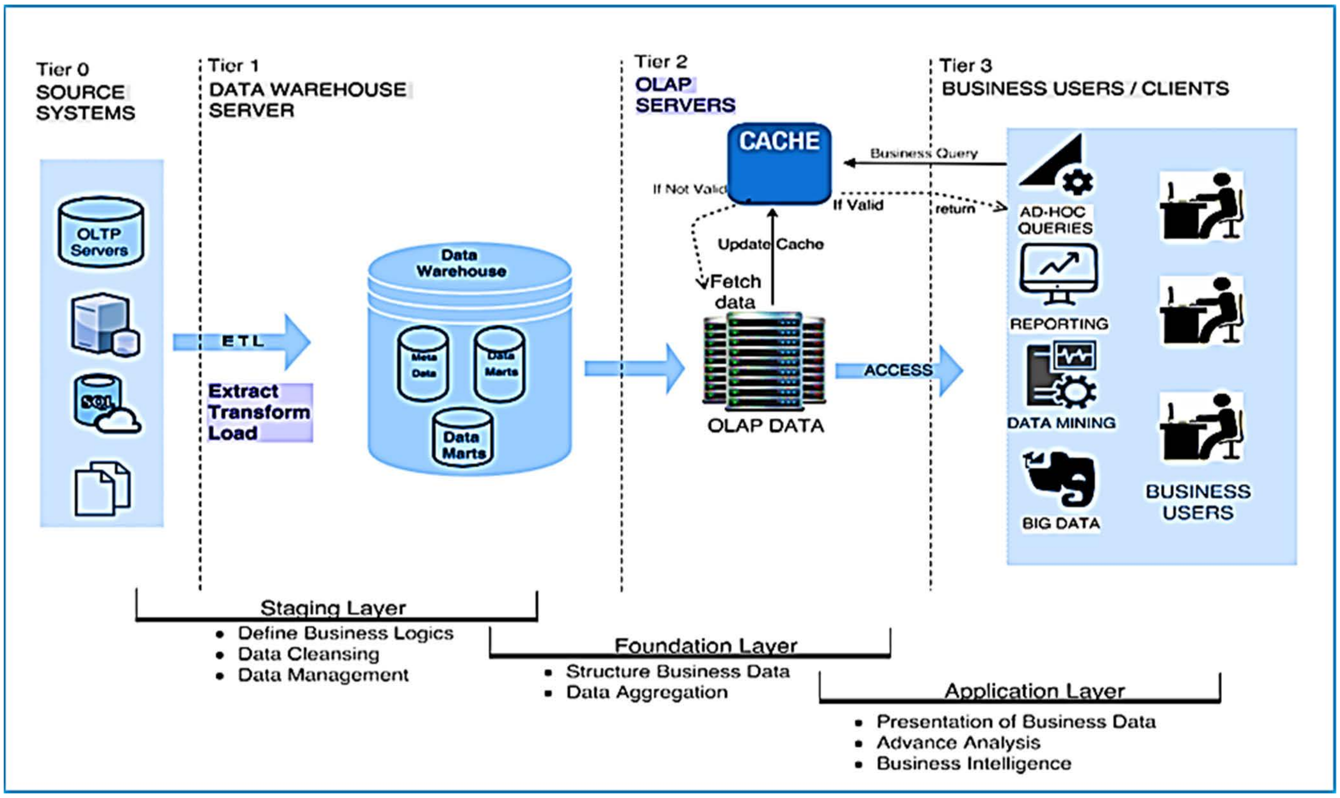

FIGURE 2. Data warehouse architecture with cache mechanism.

make this data pure to use it for a single integration point. After this step, the data is sent to the transformation step for data cleansing [27].

2) TRANSFORMATION

Data is extracted into a single database, but it is still not usable. Here comes the transformation of that data into a single format. There are various potential transformations, for example, cleansing the data, correcting misspellings, resolving domain convicts, managing with missing elements, parsing into standard formats, and reduplicating data. Transformation is the second and most important step of ETL. Furthermore, the transformation of different data needs various techniques depending upon the business need. Great data sources will require little change, while others might require one or more change strategies to meet the business and specialized prerequisites of the object database or the DW [28].

3) LOADING

The final step of the ETL process is the loading of data into OLAP. During the load step, it is important to guarantee that the load is performed accurately and with as minimum resources as possible. The loading process often targets the database. The process varies widely. Depending on the requirements of the organization, A few DW overwrite the previous information with upgraded information. This data upgrade is performed as frequently as possible on a daily, weekly, or monthly basis. This part of DW comprises the entire business data aggregation. The data is then uploaded into the OLAP system of DW to minimize the number of rows. This layer of DW is also known as the foundation layer. Now, this foundation layer interacts with the presentation or application layer for data representation [28].

V. RESULT AND DISCUSSIONS

In this section, we describe the comparisons of the replacement techniques, cache workflow, and discuss the experimental results.

A. REPLACEMENT TECHNIQUES COMPARISON

The replacement techniques or strategies Least Frequently Used (LFU), Least Recently Used (LRU) and Size used for cache memory, are focused the most to overcome the load of cache memory. The LRU, LFU, and SIZE policies are used for a long time, these techniques are combined with other techniques to make a new solution for the replacement of cache data. Table 2 shows the advantages and disadvantages of these techniques if we used them separately. To assess the performance of our approach we compare the size-based, aging frequency-based policies.

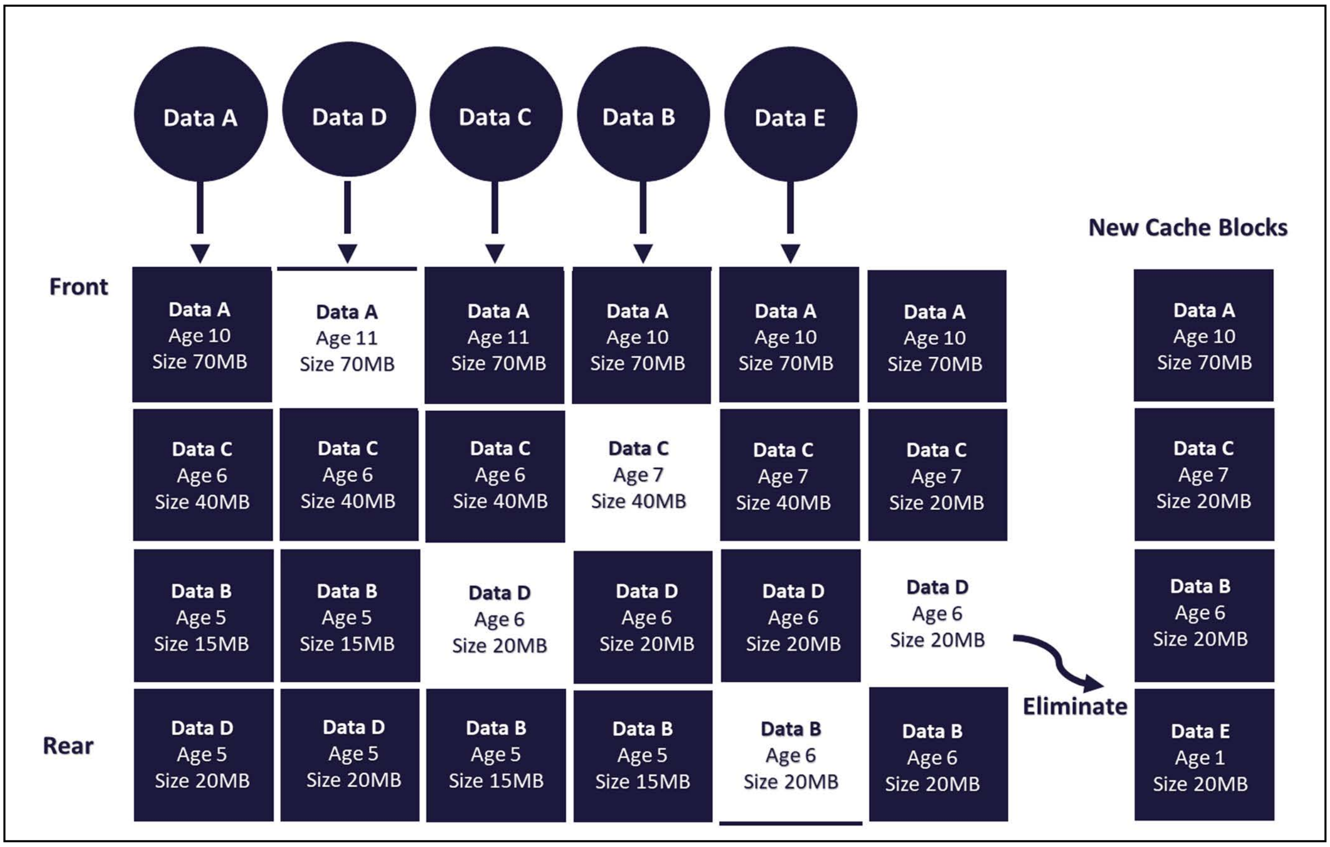

FIGURE 3. Cache data and reuse in workflow.

B. CACHE DATA AND REUSE IN WORKFLOW

The cache is a computer memory used for the storage of recently used or frequently used data, to save time when accessing a large volume of data again or when running large workflows. We can persist data in workflows while using cache. In Figure.3 the workflow shows how cache memory replaced data when new data is requested. Also shows step by step mechanism of cache to change the positions and ages of data depending upon how many times users requested the same data.

TABLE 2. Cache replacement algorithms comparison.

Policy	Description	Advantages	Disadvantages
LRU	It first removes The Least Recentl y Used objects	It works perfect and efficient with uniform data	Ignores to tackle cache overloads
LFU	It first removes the Least Frequently Used objects	Simplicity and easy implementation	Ignores cache overloads
SIZE	It first removes the data with high memory size	Overcom e cache overloads	Not suitable for all domains

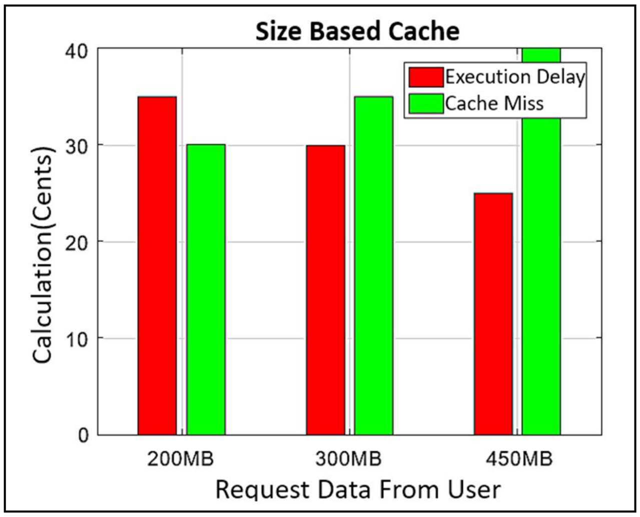

FIGURE 4. Simulations of size based technique.

C. SIMULATION RESULTS

We performed the simulation using different sizes of data. When the user requested a different size of the data, it affects

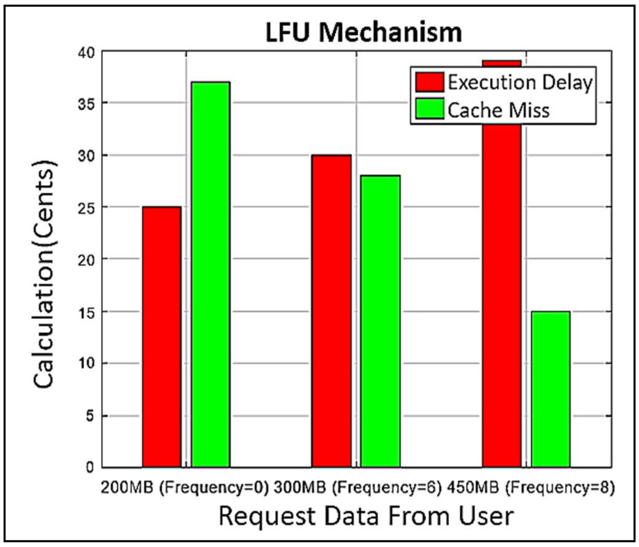

FIGURE 5. Simulations of LFU technique.

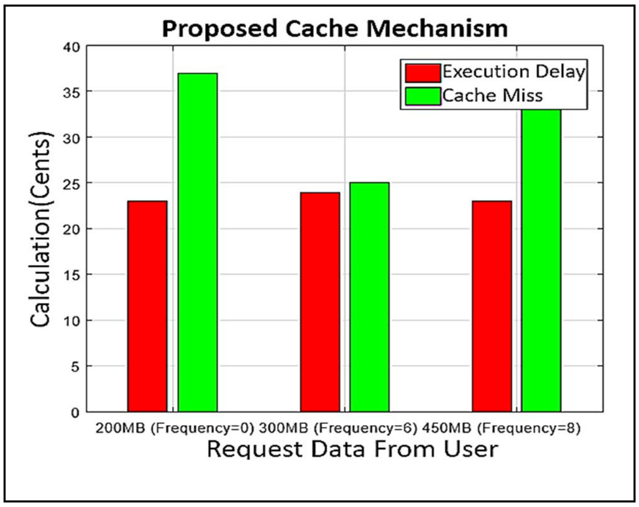

FIGURE 6. Simulations of proposed cache mechanism.

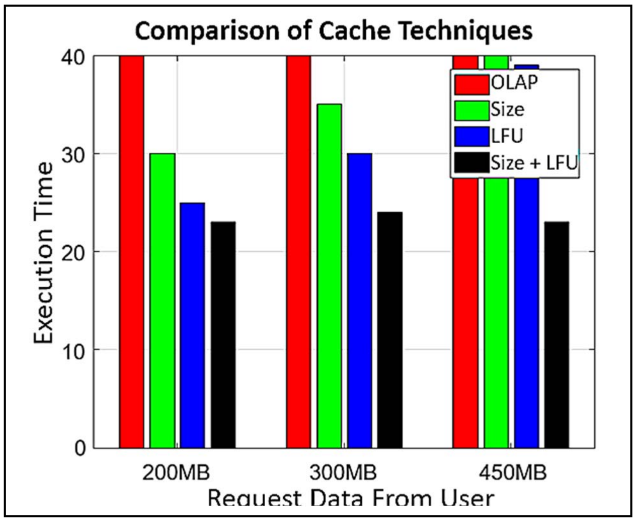

FIGURE 7. Simulations and performance comparison.

the size-based cache. In Figure. 4 simulations result of the size-based technique are shown that when the small data (200 MB) is requested the chance of the cache miss is less as compared to when a user requested for more amount of the data. So, when the size of requested data increased, then there are more chances for a cache miss. Because in size based, data that is consuming more memory size is eliminated first.

In Figure.5, simulations of the LFU technique are evaluated. As shown in Figure.5 User requested different size of data from memory and the results reveal that when the size of requested data increase, then there are more chances for a cache hit, and the execution delay is not much affected, this technique performs better in terms of execution as compared to the size base technique & OLAP.

Whereas in Figure 6, the results of our proposed technique are simulated. The user requested different sizes of data from memory. Results show that when the size of requested data increases, the execution time starts to decrease by considering the size and frequency of data to calculate fetching performance. This technique performs better as compared to the Size and LFU techniques.

Figure 7, shows the comparisons of simulations results of these techniques. LFU and Size perform better as compared to the other techniques. Results show that by hybridizing the Least Frequently Used and Size-based techniques we reveal the better result and optimize the data accessing performance.

VI. CONCLUSION AND FUTURE WORK

Using LFU, LRU, SIZE or any cache replacement technique depends upon the data environment and domain. Because of increasing data, DW implementation of cache memory is a challenging task for OLAP. We have proposed our cache mechanism by combining frequency, size, and aging-based policies. By combining all of them, it will outperform as compared to LFU, LRU, and SIZE based separately.

In the future, we are going to implement our proposed algorithm in the real-life testing scenario for the OLAP database server, and we will use materialized views and concatenation to optimize cache performance to improve analytical query response time in a real-time data warehousing environment.

REFERENCES

[1] W. Moudani, M. Hussein, M. Moukhtar, and F. Mora-Camino, "An intelligent approach to improve the performance of a data warehouse cache based on association rules," *J. Inf. Optim. Sci.*, vol. 33, no. 6, pp. 601–621, Nov. 2012.

[2] A. Simitsis, P. Vassiliadis, and T. Sellis, "Optimizing ETL processes in data warehouses," in *Proc. 21st Int. Conf. Data Eng. (ICDE)*, Tokyo, Japan, 2005, pp. 564–575.

[3] D. Matani, K. Shah, and A. Mitra, "An O (1) algorithm for implementing the LFU cache eviction scheme," Oct. 2021, *arXiv:2110.11602*.

[4] S. Huang, Q. Wei, D. Feng, J. Chen, and C. Chen, "Improving flash-based disk cache with lazy adaptive replacement," *ACM Trans. Storage*, vol. 12, no. 2, pp. 1–24, Mar. 2016.

[5] M. S. A. Khaleel, S. E. F. Osman, and H. A. N. Sirour, "Proposed ALFUR using intelegent agent comparing with LFU, LRU, SIZE and PCCIA cache replacement techniques," in *Proc. Int. Conf. Commun., Control, Comput. Electron. Eng. (ICCCCEE)*, Khartoum, Sudan, Jan. 2017, pp. 1–6.

[6] Z. Wang, K. Zeng, B. Huang, W. Chen, X. Cui, B. Wang, J. Liu, L. Fan, D. Qu, Z. Hou, T. Guan, C. Li, and J. Zhou, "Grosbeak: A data warehouse supporting resource-aware incremental computing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2020, pp. 2797–2800.

[7] F. Ravat, O. Teste, R. Tournier, and G. Zurfluh, "Algebraic and graphic languages for OLAP manipulations," *Int. J. Data Warehousing Mining*, vol. 4, no. 1, pp. 17–46, Jan. 2008.

[8] M. O. Akinde, M. H. Böhlen, T. Johnson, L. V. S. Lakshmanan, and D. Srivastava, "Efficient OLAP query processing in distributed data warehouses," *Inf. Syst.*, vol. 28, nos. 1–2, pp. 111–135, Mar. 2003.

[9] N. Gupta and S. Jolly, "Enhancing data quality at ETL stage of data warehousing," *Int. J. Data Warehousing Mining*, vol. 17, no. 1, pp. 74–91, Jan. 2021.

[10] P. Kranas, B. Kolev, O. Levchenko, E. Pacitti, P. Valduriez, R. Jiménez-Peris, and M. Patiño-Martinez, "Parallel query processing in a polystore," *Distrib. Parallel Databases*, vol. 39, pp. 1–39, Feb. 2021.

[11] M. Zhang and H. Wang, "Approximate query processing for group-by queries based on conditional generative models," 2021, *arXiv:2101.02914*.

[12] W. Lehner, "Query processing in data warehouses," in *Encyclopedia of Database Systems*. Boston, MA, USA: Springer, Jan. 2009, pp. 2219–2317, doi: 10.1007/978-0-387-39940-9_298.

[13] S. Chaudhuri and U. Dayal, "An overview of data warehousing and OLAP technology," *ACM SIGMOD Rec.*, vol. 26, no. 1, pp. 65–74, Mar. 1997.

[14] A. Becher, D. Ziener, K. Meyer-Wegener, and J. Teich, "A co-design approach for accelerated SQL query processing via FPGA-based data filtering," in *Proc. Int. Conf. Field Program. Technol. (FPT)*, Queenstown, New Zealand, Dec. 2015, pp. 192–195.

[15] A. Longo, S. Giacovelli, and M. A. Bochicchio, "Fact—Centered ETL: A proposal for speeding business analytics up," *Proc. Technol.*, vol. 16, no. 1, pp. 471–480, 2014.

[16] M. F. Masouleh, M. A. A. Kazemi, M. Alborzi, and A. T. Eshlaghy, "Optimization of ETL process in data warehouse through a combination of parallelization and shared cache memory," *Eng., Technol. Appl. Sci. Res.*, vol. 6, no. 6, pp. 1241–1244, Dec. 2016.

[17] R. Mukherjee and P. Kar, "A comparative review of data warehousing ETL tools with new trends and industry insight," in *Proc. IEEE 7th Int. Advance Comput. Conf. (IACC)*, Jan. 2017, pp. 943–948.

[18] P. Tiwari, S. Kumar, A. C. Mishra, V. Kumar, and B. Terfa, "Improved performance of data warehouse," in *Proc. Int. Conf. Inventive Commun. Comput. Technol. (ICICCT)*, Mar. 2017, pp. 94–104.

[19] Z. Chen, G. Cong, and W. G. Aref, "STAR: A cache-based distributed warehouse system for spatial data streams," in *Proc. 29th Int. Conf. Adv. Geographic Inf. Syst.*, Nov. 2021, pp. 606–615.

[20] M. R. Kaseb, S. S. Haytamy, and R. M. Badry, "Distributed query optimization strategies for cloud environment," *J. Data, Inf. Manage.*, vol. 3, no. 4, pp. 271–279, Oct. 2021, doi: 10.1007/s42488-021-00057-z.

[21] P. Michiardi, D. Carra, and S. Migliorini, "Cache-based multi-query optimization for data-intensive scalable computing frameworks," *Inf. Syst. Frontiers*, vol. 23, no. 1, pp. 35–51, Feb. 2021.

[22] P. N. Nagendra, "Improving instruction cache performance for modern processors with growing workloads," ProQuest Diss. Publishing, Princeton Univ., Princeton, NJ, USA, Tech. Rep. Nagendra_princeton_0181D_1377, Sep. 2021, pp. 1–121, vol. 28644741. [Online]. Available: http://arks.princeton.edu/ark:/88435/dsp01js956j943

[23] V. Gour, S. S. Sarangdevot, A. Sharma, V. Choudhary, R. B. Patel, and B. P. Singh, "Improve performance of data warehouse by query cache," in *Proc. AIP Conf.*, 2010, pp. 198–200.

[24] W. Zhang and K. A. Ross, "Permutation index: Exploiting data skew for improved query performance," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Dallas, TX, USA, Apr. 2020, pp. 1982–1985.

[25] A. V Hoof, "Top40 cache algorithm compared to LRU and LFU," Citeseer, Mediapark Hilversum SNE, Univ. Amsterdam, Amsterdam, The Netherlands, Feb. 2009, pp. 1–11. [Online]. Available: https://rp.os3.nl/2008-2009/p27/report.pdf

[26] N. Megiddo and D. S. Modha, "ARC: A self-tuning, low overhead replacement cache," in *Proc. FAST 2nd USENIX Conf. File Storage Technol.*, San Francisco, CA, USA, vol. 3, Apr. 2003, pp. 115–130.

[27] R. Kimball and M. Ross, *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. Hoboken, NJ, USA: Wiley, Aug. 2011, pp. 1–389.

[28] A. V. Michael and P. Ahirao, "Improved use of ETL tool for updation and creation of data warehouse from different RDBMS," in *Proc. 3rd Int. Conf. Adv. Sci. Technol. (ICAST)*, Apr. 2020, pp. 1–4, doi: 10.2139/ssrn.3565505.

CH ANWAR UL HASSAN (Member, IEEE) received the B.S. degree in software engineering from the National University of Modern Languages, Islamabad, Pakistan, in 2015, and the M.S. degree in software engineering from COMSATS University, Islamabad, in 2019.

After Graduation, he was working as a Visiting Lecturer with the Federal Urdu University of Arts, Science and Technology, Islamabad. He is currently working as a Lecturer with the Capital University of Science and Technology, Islamabad. He is an Experienced Software Engineer, from 2015 to 2017, where he was working as a Web/App Developer and was involved in the development of different top-notch software being involved from documentation to deployment. From 2017 to 2019, he was working as a Research Assistant at COMSATS University, Islamabad. His research interests include blockchain, data warehousing, data analysis, machine learning, data mining, smart grid energy management, software process improvements, and software costing and estimation

Mr. Anwar's awards and honors include the multiple IBM Certifications, Debate Competition Winner and Runner-Up Awards and Research Poster Winner Certificate at COMSATS University, in 2017.

MUHAMMAD HAMMAD received the B.S. degree in software engineering from International Islamic University, Islamabad, Pakistan, in 2015, and the M.S. degree in software engineering from FAST-NUCES University, Islamabad, in 2018.

After Graduation, he was working as a Visiting Lecturer at Quaid e Azam University, Islamabad. He is currently working as a Lecturer with the Capital University of Science and Technology, Islamabad. He is an Experienced Software Engineer in the IT industry with more than 3.5 years of experience in full-stack development and research work and involved in the development of different top-notch software being involved from documentation to deployment. He has major expertise in development at Microsoft Technologies (Asp.net and MS SQL Server), angular framework, agile development (Scrum, XP, and Kanban), database development, and software configuration management (TFS and GIT).

MUEEN UDDIN (Senior Member, IEEE) received the B.S. and M.S. degrees in computer science from Isra University Hyderabad, Pakistan, and the Ph.D. degree from Universiti Teknologi Malaysia (UTM), in 2013. He is currently working as an Assistant Professor of cybersecurity and blockchain with Universiti Brunei Darussalam. He has authored more than 100 international research articles published in highly indexed and reputed journals. His research interests include blockchain, cybersecurity, cloud computing, and virtualization.

JAWAID IQBAL received the Ph.D. degree from Hazara University, Mansehra, in 2021. He has been teaching at the university level for more than nine years. He started his career at the IT Department, Hazara University, in 2013. He also served at different universities, like the Abbottabad University of Science and Technology (AUST) and the University of Sialkot. Currently, he is an Assistant Professor with the Department of Computer Science, Capital University of Science and Technology, Islamabad. He has taught various subjects of computer science at bachelor's and M.S. program levels. He is a member of the Advance Network and Security Research Group, CUST. He has numerous publications in international conferences and journals. His research interests include information security and networks. He has intention to work with people who love to practice new ideas, where he can excel by utilizing his potential and broaden his horizon by complementing the theoretical knowledge with practical experience of the professional life, experiencing the culture of teamwork and individual excellence along the way.

JAWAD SAHI received the B.S. degree in computer science from COMSATS University, Islamabad, Pakistan. He is currently pursing the M.S. degree in software engineering form the Department of Computer Science, COMSATS University.

He is an Experienced Software Developer in the IT industry. He worked as a Data Analyst at Tera Data, Islamabad, Pakistan. He has major expertise in data analysis and digital marketing. His research interests include data analysis, data warehousing, smart grid, and energy management.

SADDAM HUSSAIN received the bachelor's degree from the Islamia College, Peshawar, in 2017, and the master's degree from Hazara University Masehra, Pakistan, in 2021. He is currently pursuing the Ph.D. degree with the School of Digital Science, Universiti Brunei Darussalam, Brunei. He has published several papers in well-reputed journals, including IEEE, *JISA* (Elsevier), *Cluster Computing*, *Computer Communication*, IEEE INTERNET OF THINGS JOURNAL, Hindawi, *CMC*, and *Electronics*. His research interests include cryptography, network security, wireless sensor networking (WSN), information-centric networking (ICN), named data networking (NDN), smart grid, the Internet of Things (IoT), the IIoT, quantum computing, cloud computing, and edge computing. He is serving as a Reviewer in reputed journals, including IEEE ACCESS, *International Journal of Wireless Information Networks*, *Scientific Journal of Electrical, Computer, and Informatics Engineering*, and *CMC*.

SYED SAJID ULLAH received the master's (M.S.) degree in computer science from Hazara University Mansehra, Pakistan. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Villanova University, PA, USA. He is currently working as a Researcher with the National Institute of Standards and Technology (NIST) in the projects (Practical implementation of Quantum Cryptography and Security Solutions for Future Internet Architecture Named Data Networking). His research interests include cryptography, network security, information-centric networking (ICN), named data networking (NDN), and the IoT.