Data Warehouse Performance
Special Topics Report - Team 4

Advisor
Prof. Weider Yu
CMPE, SJSU, CA, USA
weider.yu@sjsu.edu

Manjushree Berike Rajanna
CMPE, SJSU, CA, USA
SID - 015275377
+18582827828
manjushree.berikerajanna@sjsu.edu

Sonali Thote
CMPE, SJSU, CA, USA
SID - 015357524
+16504319276
sonali.thote@sjsu.edu

Shruti Savardekar
CMPE, SJSU, CA, USA
SID - 015411396
+14082075334
shruti.savardekar@sjsu.edu

Priyanka NAM
CMPE, SJSU, CA, USA
SID - 015287805
+19098457456
alivelumanjulokapriyanka.nareddy@su.edu

Sakruthi Avirineni
CMPE, SJSU, CA, USA
SID - 016009929
+1 6693699968
sakruthi.avirineni@sjsu.edu

***Abstract*—One of the most crucial IT-based systems in enterprises is the business intelligence (BI) system Data Warehouse (DW). The character and performance of an organization may be significantly impacted by the decisions made utilizing these technologies. Data is initially imported into the data warehouse system before being retrieved. As compared to transactional data, a data warehouse frequently contains orders of magnitude more data. Complex queries must be run to do business intelligence on a data warehouse system. In order to maximize the performance of the business intelligence system, performance is crucial. A data warehouse's performance may be increased by doing a number of different actions.Precomputation of materialized views is one of the steps, among others. In this essay, we'll talk about various techniques for enhancing Data Warehouse performance. Query optimization is a significant component that affects the performance of data warehouses. Fundamentally, there are a number of approaches that may be used to optimize SQL/NoSQL queries before they are employed on enormous databases, including strategic queries, query caching, locking unlocking procedures, query management, etc. Such strategies will be covered in great depth in this essay.**

Keywords—Performance, Query Cache, Joint-Free Approach, Join Module, FuzzyHive, Neutral Data.

I. Introduction

Any large-scale business has to have a strong business intelligence department. Companies are able to get a huge quantity of data from their consumers because of the growing usage of computers, cellphones, and the internet. Such unprocessed data may not be accurate for the business unless some intelligence is developed. Business intelligence is the process in question. Customers' transactional data is frequently needed to be kept in a data warehouse system since it offers the best means of extracting insight from it. A variety of experiments may be run once the data has been cleaned, normalized, and stored in the data warehouse. The OLTP system records transactional data produced by the customers. Online transaction processing, or OLTP, is another name for it. Where rows and columns make up the structural format of the data storage. The data is saved in some sequence, such as a timestamp, and each data input is distinct. The OLAP system houses the analytical data utilized for business intelligence. The term "OLAP" stands for "online analytical processing." In contrast to OLTP, which has a flat structure, OLAP has a columnar structure. A whole history of records is maintained in a columnar format, similar to OLAP. OLAP transactions are intended to be supported by the data warehouse system. Data warehouses are not designed to offer low latency processing and quick processing. The support for complicated queries is the primary focus instead. To extract intelligent information from enormous amounts of raw data, complex queries are needed.

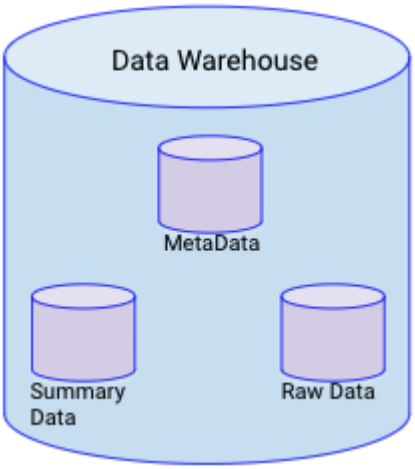

Figure 1. Data Warehouse

Data Profiling is a crucial feature that is used to clean the data once it has been stored in the Data Warehouse. Data profiling is the process of assessing, testing, and analyzing the data in order to produce insightful summaries or analytics. Data profiling in data warehouses may be done using a variety of analytical methods. Minimum, maximum, average, percentile, entity-relationship, and frequency are examples of these algorithms. An organization can understand patterns from the data once data profiling is done on the data from the data warehouse.

Performance is of the highest importance in a data warehouse system. The environment data is far larger than the OLTP configuration, just as it is in OLAP. There are two ways to improve the performance of the data warehouse by scaling: vertically or horizontally.

Data warehousing has evolved in a way that consistently sacrifices commercial

value in order to get around technological obstacles. The automated use of summary data or data models designed for a specific application is one example of a compromise.

At first, businesses merely took data from operational systems and stored it in a different database, often the one used for processing online transactions. This prevented more time-consuming and complicated analysis workloads from interfering with OLTP work that was more mission-critical.

The majority of operational systems store data using a normalized model, which reduces duplication and complicates data linkages. While helpful for the OLTP workload, this architecture may prevent the OLTP databases in question from efficiently handling analytical queries.

Another significant element that is crucial to the optimization of database system queries. Depending on the requirements, the database system may be either SQL, NoSQL, or a combination of the two. BI systems will put a significant amount of demand on the database system to get information, regardless of the database structure. Every query has the potential to use system memory and database time resources. Consequently, it is crucial to optimize those queries. This article will go into great length on a set of rules and procedures that may greatly enhance query performance and help to optimize the operation of the data warehouse.

II. Related Work

As "Big Data" becomes more important, NoSQL databases are becoming more and more popular. [17] compares the performance of many NoSQL and SQL databases using key/value stores and finds that not all NoSQL databases perform better than SQL databases. Every read or write operation affects a database's performance, and there is only a weak correlation between performance and the data model it uses. However, there is currently no NoSQL real-time DW solution for stream joins. Both SQL Server and MongoDB go through a number of stress tests as part of the study [19] that examines the response speeds of these two types of databases. Linux operating systems are outperformed in terms of performance by MongoDB, however SQL Server is slower when handling large amounts of data. The focus of this study, however, is not on SQL and NoSQL database stream join methods.

This research [16] compared the performance of the NoSQL database to the conventional SQL database for processing a limited amount of structured data using the MongoDB and SQL server databases. Unless aggregating techniques are applied, the authors claim that MongoDB outperforms relational databases for growing databases. The stream join techniques of MongoDB and their SQL equivalents for real-time data warehousing are not compared in this study.

When dealing with the massive volumes of big data created by the web that are not organized, traditional RDBMS are inadequate. On the other hand, NoSQL is capable of handling the data types mentioned in [15] and [14].

The study in [18] on NoSQL ETL is crucial because it is challenging to assess and make business decisions based on unstructured data because conventional data warehouses only store structured data. The ETL phase join technique to be used for unstructured data streams in real-time data warehousing will be determined by the results of this research. [15] develops and evaluates DW ideas based on NoSQL. The authors contend that building a NoSQL DW is only feasible if an appropriate compromise is made between the traits of a SQL DW and the capabilities of a NoSQL DBMS. The ETL phase NoSQL stream connectors necessary for the installation of NoSQL-based DW are not covered by this study.

Pre-join has been used to expedite the execution of database queries. In [8, 14], the universal relation is utilized to load all data into the fact table, allowing for the sequential processing of queries. However, this strategy requires a lot of storage space. Unlike the universal relation, our method shifts the residual joins to the post-processing of the aggregate results and just compresses the key hierarchy information of the dimension tables into the surrogate keys of the fact table. A decent trade-off between storage and joint costs should come from this. Join-index [15], a different kind of pre-join, materializes the addresses of the join keys in the foreign key table. The dimension tables must be frequently queried in order to retrieve the join results when using join-index to perform star queries, adding expensive random visits. In contrast, our method materializes the join key values and their parent hierarchies in the fact tables, enabling us to provide the join results in a single scan.

Users increasingly seek information that is more current as well as more complicated and flexible. Data must be accessible around-the-clock, and many business users expect that decision-supporting data be available within a short period of time—in some circumstances, minutes or even seconds—after an event occurs.

The demands of transferring data across many platforms, modifying and preparing response sets, and being aware of the inquiry in advance have turned into a barrier to corporate expansion. Developers must understand how to leverage data warehousing strategies to offer solutions that streamline systems so they need less data transfer and propagation, as opposed to complicating the architecture.

III. Proposed Methodology

There are several methods for enhancing the performance of a Data Warehouse, as mentioned in the previous Related Works section. This will help us narrow down a few approaches and compare them side by side to see which is best suited for a

particular kind of data. It is not advisable to move from a detailed level to a summary one without fully understanding the implications. There are not many separate phases in the procedure. The capacity of your technology to support your ongoing company demands should be severely reconsidered if it is not able to support commencing at the most basic level.

A. Query Cache

Maintenance is the biggest issue with data warehouses since adding data is simple but changing it is challenging. It holds a copy of the transaction that is typically read-only and that the transactional system cannot alter. Large amounts of data are typically extracted from a data warehouse in order to produce reports, graphs, charts, etc. Therefore, it may be used to store only the data necessary for data analysis rather than maintaining all the data. Consequently, a variety of methods are available to examine the functionality and turnaround time of such a massive data storage. The Query management system is one of them. Instead of using the entire data set, the searches in this case will only function on summary data.

This approach works by breaking down large queries into smaller ones, and each smaller query is then executed on a separate processing core. This makes it possible for each query to receive enough processing time and produce quicker responses. It takes a certain route to reduce cross-network data retrieval and data from various sources.

This approach is still not the best for efficiently running queries and obtaining data, though. A Query cache may yet be added to the system's implementation to make it better. The query cache stores the most recent searches and their results, operating similarly to a cache. When a user issues the same query again later, the system will quickly retrieve results from the cache. However, if data is grabbed from the cache when a user executes an insertion, deletion, or update query, the results may be incorrect. In order to distinguish between results that are legitimate and those that are invalid, a flag is kept in the query cache. For instance, suppose a user tries to insert some data while there is already data in the cache. The flag in the cache will then be flagged as invalid before the insert query is executed. The system is aware that there is faulty data in the cache and will not utilize it the next time a fetch query is triggered.

The materialized views must also be updated when a warehouse is modified, particularly when distant information sources change. The time window for updating the warehouse is getting smaller even though there are more queries asking for up-to-date data and more data being mirrored to data warehouses. An effective view maintenance approach is therefore one of the unresolved problems in the data warehouse environment. This lessens OLAP query interference and downtime

while improving query processing performance.

Let's now get into more detail about the query cache's mechanism. The query cache's flag lets you know if the data it contains is legitimate or not. These two words signify:

- Valid – The state will be valid if a query is kept in the query cache and not modified in the database as a consequence of any of these query insert, update, or delete actions.
- Invalid - The status is invalid if the query is not saved in the query cache. The state will be invalid if the user updates the data using any of these queries (insert, update, or delete).

Our problem is that a query and its response are already in the cache. The cache query result will, however, show up in the outdated data if the warehouse is updated with fresh information. We'll create a state machine.

- Write query management - The cache redirects queries to insert them in cached temporary tables when an update, insert, or delete query is received. The query cache sends UPDATE and DELETE requests to both the actual tables in the database and the temporary tables. Temporary tables in the query cache later change the original in the database. Performance will thus increase as a result of the speedy retrieval of the needed data via cache queries.
- Read query management - If a user submits a query and the outcome is not recorded in the query cache, the state is invalidated, and the outcome is fetched from the database and shown. The status is valid and the result is presented from the query cache if a user submits a query and the result is saved in the query cache. There is no longer a requirement for the query to search through every document. The query cache's most recent index of the query result will be retrieved. Then, it will start looking through that index for records that meet the query requirements. This can help you save a lot of time and effort compared to manually searching through a lot of data.

The first research is only the beginning of the data warehouse lifecycle, which goes on indefinitely until the demand is satisfied. The most recent queries that were run will be recorded in the query cache. The results of recently finished searches will likewise be kept in the query cache. Queries and the results they return are stored using the Query Cache method. If another user performs a similar inquiry, cache memory will be used to get the answer. When responding to the outcomes of prior user requests, this memory will be utilized. This can help you save a lot of time and effort compared to manually searching through a lot of data.

B. Joint-Free Approach

Typically, a data analytics platform normalizes and organizes the data in the data warehouse into fact tables and dimension tables before processing or accessing it. Then, it connects these tables throughout the data retrieval process and displays the outcomes. The star schema is a method for handling the queries in a data warehouse. However, because the data is so large and joining processes take so much time, this strategy has become conventional and is no longer very effective. Additionally, we cannot reuse the prior joins when using the star schema and must execute repeated joins. As a result, we will now investigate a different architecture for data warehouse query processing. This framework is incredibly quick, dependable, and scalable.

C. Join Module

Real-time data warehouses (DW) may monitor performance by using the Join module, which is a part of the Extracting, Transforming, and Loading (ETL) sublayer. To compare the processing speeds of SQL and NoSQL systems, it is necessary to look at how well the connect module works with both SQL data and NoSQL data. Only after completing this step can the comparison be done. Building six fictitious datasets in addition to the one real dataset, each with its own size and categorization, is necessary to take advantage of stream processing. The synthetic and real-world datasets must each be produced as a single instance. The ids of the datasets that were utilized to create each stream must be included in the stream file. The file has to contain these ids. A PC with an Intel Core i5 CPU operating at 1.70 GHz and 4 GB of RAM is required for the successful deployment of MongoDB Compass with MongoDB Server version 3.0.

D. FuzzyHive Module

Reporting and querying from huge amounts of structured, semi-structured, and unstructured data frequently calls for some flexibility. Fuzzy sets' flexibility makes it possible to classify the outside world in a fluid, human-mind-like way. A data warehousing system called Apache Hive is built on top of the Hadoop big data processing platform. Hive enables the execution of queries as well as the aggregation and analysis of data held in other repositories as well as the Hadoop distributed file system.

E. Neutral Data Model

Utilizing atomic data in a normalized model that is general to all functions and groups constitutes the lowest level. The data has the minimal amount of detail required to support the business goals [20]. Although it might not always include transaction data from the beginning, it should be possible to descend to that level when requirements evolve.

Most businesses define the detail tables using access views for concurrency and security reasons. Access views have a locking modification called a "read without intent" or "dirty read" that enables users to choose data from the table even as

others are writing to it. After that, queries are executed against those views rather than the actual database, which helps to resolve many locking issues. As long as they accurately reflect the underlying detail tables, access views are permitted.

Run an optimized Structured Query Language (SQL) query on the detail data to find out if the desired performance is feasible in many cases. Poor performance often results from the tool or user not executing the most effective statements. This kind of structure is used by many prosperous firms, and it is maybe the reason why they are prosperous in the first place.

F. Implemented Views

Table views should be implemented in order to improve performance and make navigating easier. Although views normally have little impact on performance, employing views may result in the tool producing "better" SQL when a third party demands a certain data model [20]. You do not need to rely on each front-end tool vendor to have in-depth knowledge of every database on the market because views may be tuned for a certain database.

Additionally, you may include optimum join strategies in the views rather than having that every tool and user be knowledgeable about them. This method also strengthens the security of the data tables; for instance, a telecommunications business employed views to build a logical star schema model for a specific tool. The query tool "perceived" the data model as a star schema even though the star model was not physically existent because the joins required by a normalized model were concealed in the view specification for a model particular to the tool.

By using views in this way, the database administrator may provide each business unit its own logical functionalization. Even though they define the same element differently, views display data to users in the manner in which they anticipate seeing it. For instance, one department could prefer data presented in feet, while another would prefer it presented in inches. Data may be transformed into a logical view definition and only need to be saved once when utilizing views. This reduces data duplication, guarantees consistency, and streamlines data administration duties.

G.Add Indexes

The next step towards improvising performance is to add indexes, which can be as basic as secondary indexes or as complex as covered, join, or aggregate indexes. Usually, adding indexes will suffice to satisfy the performance requirement. The key benefit of indexes is that the system keeps them updated at the same time as the basic data tables [20]. The answer sets and the detailed data will always agree, notwithstanding the expense associated with maintaining and automatically updating indexes. In smaller reference tables, comparing against the detailed data is a common practice, but

using this method to bigger, more regularly updated tables calls for careful thought.

H.Expand the System

If indexes and views are unable to address performance difficulties, you will need to reevaluate your data management approach and decide what your long-term business needs are. You must enquire two things:

- What kind of expansion would be required to satisfy the performance criterion?
- What advantages for your firm would you get from fulfilling that requirement?

The addition of downstream operations, such as summary tables and further data extraction or processing, may hinder the achievement of your real-time data warehouse aim. Increasing the system configuration is a possibility if the expense can be justified [20]. However, the competence of your data- base will play a major role in this process. For various operations like joins, sorting, or aggregations, many databases include serial processing.

If your database includes serial processing points, increasing the system might not improve performance since it won't speed up the query process at that point.

Consider additional options that necessitate retaining the extracted data independently from the detailed data if extending the system is not technically possible or economically reasonable. These options also bring up new factors including price, storage space, data management, and time delays. One merchant, for instance, favors a single data repository with no downstream processing. This merchant made the decision that it was preferable to receive a two-minute response to a question a few hours after a sale than a sub second response a week after the transaction.

IV. Results

Examining the results. Transform, aggregate, merge, and post-processing are all combined into one stage in the data warehouse method as described. Hierarchy encoding is used to condense the information about the dimension hierarchies into the fact table, which avoids the requirement for star (snowflake) joins during query processing. Because of this, it is possible to filter and aggregate the fact table without using the dimension tables, and the number of joins has no bearing on the query execution time. A data warehouse is a repository of integrated and consistent data that is meant to support business analysts and decision-makers. By adding new data from multiple sources and maintaining updates after a predetermined amount of time, the warehouse may be updated.

The first study is the first step in the data warehouse life cycle, which goes on indefinitely until the demand is satisfied. The most recent queries that were run will be recorded in the query cache.

The list of several methods to boost a data warehouse system's performance from the study is summarized below. The procedure known as "query caching" involves saving some of the data on the cache rather than the database. As is common knowledge, reading data from a cache is almost ten times quicker than reading data from a disk. Thus, storing the data that is utilized the most frequently in the cache results in a noticeable speed boost. With the joint free method, fewer joins are required between tables. It can be a time-consuming and resource-intensive operation to combine data from various columns using a join statement. Thus, it is advised to denormalize the data in a data warehouse architecture without using join statements.

To take these acts, one must be aware of both the costs and rewards associated with each step. Making choices that support both short- and long-term requirements is also necessary. In some circumstances, you may add denormalized data models or build summary tables, which you would ultimately remove as the functions changed over time. As long as the removal of the tables does not result in disruptions or significant application modifications, this is okay. Making a cost-benefit analysis and figuring out an enterprise's suitable degree of performance optimization are two ways to decide how far to go. Business goals and end-user needs are other factors. Users must be aware of the technological repercussions of expecting instantaneous answers to all questions. Outlining the expenses related to making and maintaining the tables is the best method to get this information.

V Conclusion

Any Business Intelligence system is built on a data warehouse system. The effectiveness of the data warehouse has a direct impact on the effectiveness of the team developing the business intelligence product and its users. In this article, we covered a number of methods for enhancing the data warehouse system's performance.

Business intelligence tools are advised to use big data databases like Apache Hive rather than conventional databases like MySQL. These big data systems are made to run as efficiently as possible given the vast amounts of data and sophisticated queries they include. The set of approaches to be used in the data warehouse must be decided by the team as a whole. Every strategy has a cost; one that increases speed may put a strain on storage, and vice versa. The team must discuss these trade-offs and choose what is best for the company. The majority of data warehouse initiatives emphasize immediately becoming online. In the haste to offer immediate value, analysis and flexibility's longer-term objectives are neglected. Many immediately go to summary tables and functional models in order to achieve the aim of having phase one satisfy urgent demands. But by spending a very short amount of time on it and laying the right

foundation, the data warehouse may have a better future.

The users, on the other side, must choose which capacity requirements must be met in order to create commercial potential. Despite the technical reason, utilizing simply summary tables will not work if the required analysis calls for detailed data. The least intrusive procedure must be used to satisfy the needs after users and implementers collaborate to understand the business requirements and their motivations. The environment becomes less adaptable to fulfilling future demands the more you shift or alter the data into response sets.

VI References

[1] Kevin S Goff, “The Baker’s Dozen: 13 Tips for Better Extract/Transform/Load (ETL) Practices in Data Warehousing”, CODE Magazine: 2017 - September/October, https://www.codemag.com/article/1709051/The-Baker

[2] C.A.U.Hassanetal.,”OptimizingthePerformanceofDataWarehouse by Query Cache Mechanism,” in IEEE Access, vol. 10, pp. 13472-13480, 2022, doi: 10.1109/ACCESS.2022.3148131.

[3] F. Di Tria, E. Lefons, and F. Tangorra, “Evaluation of Data Warehouse Design Methodologies in the Context of Big Data,” in Concurrency and Computation: Practice and Experience, vol. 28, no. 15, 2017, pp. 3–18.

[4] M. Y. Santos et al., “A Big Data Analytics Architecture for Industry 4.0,” Springer, Cham, 2017, pp. 175–184.

[5] N. Marz and J. Warren, Principles and best practices of scalable real- time data systems. Manning Publications Co., 2015.

[6] N. Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 6, Reference Architecture (Technical Report No. NIST SP 1500-6),” Gaithersburg, MD, Oct. 2015.

[7] 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.

[8] A. Simitsis, P. Vassiliadis and T. Sellis, ”Optimizing ETL processes in data warehouses”, Proc. 21st Int. Conf. Data Eng. (ICDE), pp. 564-575, 2005.

[9] S.Huang,Q.Wei,D.Feng,J.ChenandC.Chen,”Improvingflash-based disk cache with lazy adaptive replacement”, ACM Trans. Storage, vol. 12, no. 2, pp. 1-24, Mar. 2016.

[10] 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”, Proc. Int. Conf. Commun. Control Comput. Electron. Eng. (ICCCCEE), pp. 1-6, Jan. 2017.

[11] M. O. Akinde, M. H. Bo ̈hlen, T. Johnson, L. V. S. Lakshmanan and D. Srivastava, ”Efficient OLAP query processing in distributed data warehouses”, Inf. Syst., vol. 28, no. 1, pp. 111-135, Mar. 2003.

[12] 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.

[13] E. Mehmood and T. Anees, ”Performance Analysis of Not Only SQL Semi-Stream Join Using MongoDB for Real-Time Data Warehousing,” in IEEE Access, vol. 7, pp. 134215-134225, 2019, doi: 10.1109/AC- CESS.2019.2941925.

[14] R. Yangui, A. Nabli and F. Gargouri, ”Etl based framework for NoSQL warehousing”, Proc. Eur. Medit. Middle Eastern Conf. Inf. Syst, pp. 40-53, 2017.

[15] Z. Bicevska and I. Oditis, ”Towards NoSQL-based data warehouse solutions”, Procedia Comput. Sci., vol. 104, pp. 104-111, 2017.

[16] Z. Parker, S. Poe and S. V. Vrbsky, ”Comparing NoSQL MongoDB to an SQL db”, Proc. 51st ACM Southeast Conf, pp. 5, 2013.

[17] Y. Li and S. Manoharan, ”A performance comparison of SQL and NoSQL databases”, Proc. IEEE Pacific Rim Conf. Commun. Comput. signal Process. (PACRIM), pp. 15-19, Aug. 2013.

[18] D. Sahiet and P. D. Asanka, ”ETL framework design for NoSQL databases in dataware housing”, Int. J. Res. Comput. Appl. Robot., vol. 3, pp. 67-75, Nov. 2015.

[19] A. Flores, S. Ram ́ırez, R. Toasa, J. Vargas, R. Urvina-Barrionuevo and J. M. Lavin, ”Performance evaluation of NoSQL and SQL queries in response time for the e-government”, Proc. Int. Conf. eDemocracy eGovernment (ICEDEG), pp. 257-262, Apr. 2018.

[20] R. Armstrong, "Seven steps to optimizing data warehouse performance," in Computer, vol. 34, no. 12, pp. 76-79, Dec. 2001, doi: 10.1109/2.970580.