Enhancing Oracle Database Performance via Access-Pattern-Based Data Archiving

Undergraduate Research Experience Award (UREAP)

Roods Bensly Pierre

Supervised By

Dr. Mridula Sharma

Dr. Vijal Jain

May 2024

Contents

Abstract

Database management is a complex process that requires a large amount of data to be archived. In today's data-intensive landscape, efficient management of stale data poses a significant challenge for organizations. Traditional data archiving approaches rely on static rules or manual interventions. This paper presents the findings of a research project aimed at addressing a critical gap in the domain of database management, focusing on Oracle databases.

Through the implementation of a novel and dynamic data archiving solution based on access patterns and periodic processes, this research demonstrates tangible improvements in database performance and storage efficiency while minimizing manual intervention. Drawing on insights from existing literature and employing an interdisciplinary approach, the findings of this research underscore the transformative potential of access-pattern-based data archiving in Oracle databases, which contributes to advancing the discourse on database management practices.

1 Introduction

The exponential growth of data in contemporary organizational environments has led to an urgent need for efficient data management strategies. Among the myriad challenges faced by enterprises, the management of stale data within databases emerges as a critical bottleneck, particularly in the context of Oracle databases. Stale data, which remains dormant or unused over time, not only occupies valuable storage space but also hampers database performance, leading to increased query response times and decreased system efficiency.

This research project endeavors to address this pressing issue by introducing a pioneering data archiving solution tailored specifically for Oracle databases. Unlike conventional archiving approaches that rely on static rules or manual interventions, this proposed solution leverages access patterns and periodic processes to dynamically identify and archive stale data. By aligning with user query results and implementing a scheduled archiving process, we aim to optimize database performance and storage efficiency while minimizing manual intervention.

Drawing inspiration from a diverse array of literature, including works on database archiving, preservation formats, standardized archival techniques, and data cleaning, our research adopts an interdisciplinary approach. By synthesizing insights from these disparate domains, we seek to push the boundaries of traditional database management practices and pave the way for more agile and responsive data archiving methodologies.

The significance of our research extends beyond theoretical contributions to directly address real-world database performance challenges. Through the optimization of data storage, reduction of retrieval times, and enhancement of database performance, our approach promises to revolutionize how organizations manage their data, ushering in a new era of efficiency and effectiveness in database management.

2 Literature Review

The domain of database management confronts a significant challenge as organizations grapple with ever-expanding volumes of data. While traditional optimization strategies such as database partitioning (Smith, 2019) and data compression (Johnson & Brown, 2020) offer valuable insights, a critical gap remains in the dynamic archiving of stale data.

Limited research has ventured into dynamically identifying and archiving stale data within Oracle databases. Existing approaches often rely on manual or rule-based archiving, lacking adaptability to dynamic database environments. This underscores the urgent need for proactive, user-driven data archiving solutions.

Previous works discuss database archiving techniques, emphasizing the importance of systematically managing and preserving data. The development of preservation formats and archiving tools for long-term data preservation highlights the need for standardized practices. Olson (2009) discusses database archiving techniques, highlighting the importance of systematically managing and preserving data. Cha, Choi, and Lee (2015) delve into the development of preservation formats and archiving tools for long-term data preservation, emphasizing the need for standardized practices. Appuswamy (2022) presents a passive, migration-free approach to long-term database archival, offering an innovative perspective on database archiving. Additionally, Herbst (1994) explores data archiving in EXPRESS/SDAI databases, focusing on archiving data within specific database schemas.

In the context of data cleaning and optimization, Milani, Zheng, and Chiang (2019) introduce "CurrentClean," a spatio-temporal data cleaning method that addresses the management of stale data. Their work sheds light on the significance of addressing stale data in the context of database management. Williams, Li, and Olfman (2021) introduce a novel tool, DRT, for data archiving, showcasing innovative approaches to data management and archiving.

This interdisciplinary approach forms the basis of our innovative data archiving solution, positioning this research to contribute significantly to the ongoing conversation in the field of database management.

3 Research Methodology

The research methodology integrates principles from various sources to address the research objectives. This study is structured into three main stages: Data Access Monitoring, Data Archiving, and Performance Evaluation. Each stage is designed to contribute to a comprehensive understanding of data usage patterns, optimize storage, and evaluate performance impacts. Each step is crucial to the overall goal of optimizing database performance and storage efficiency.

3.1 Data Access Monitoring

The main aim of data monitoring is to identify which data is being accessed. A script is developed to establish a connection to the Oracle database and monitor data access. For a given table being monitored, an algorithm is executed periodically to go through the DBMS Audit Trail and extract the SQL TEXT and Timestamps of all the SELECT commands executed on the table. These commands are then stored in a temporary audit trail table for further processing.

Algorithm 1 Process Audit Trail

```
1: Input: audit_trail table
2: Output: Processed audit entries
3: procedure PROCESS_AUDIT_TRAIL
4:     Clear existing records from audit_trail table
5:     Insert audited SELECT queries into audit_trail table from dba_audit_trail
6:     for all row in dba_audit_trail where action_name is 'SELECT' and
7:         certain conditions are met do
8:         if SQL text doesn't contain comments or hints then
9:             Insert SQL text and timestamp into audit_trail table
10:        end if
11:    end for
12:    Call the procedure to process the audit queries ▷ process_audit_queries;
13: end procedure
```

3.1.1 How is accessed data identified?

To identify the accessed data, the algorithm processes each query in the temporary audit trail. The query is executed to extract the primary key values returned by that query. If the table associated with the query has a foreign key, the foreign key values are also extracted, as they are considered accessed.

The values from each query are accumulated into primary key and foreign key lists. These lists of accessed keys are then inserted into an "Accessed" table along with the table name from which they were extracted.

In addition, a trigger is added to the given table to capture any new data being inserted or updated. This trigger captures the primary keys of the inserted or updated rows and adds them to the "Accessed" table, ensuring that all CRUD (Create, Read, Update, Delete) operations are accounted for.

This collected data is subsequently analyzed to identify patterns of data access, which helps in determining which data is frequently accessed and which remain stagnant for prolonged periods.

Algorithm 2 Process Audit Queries Procedure

```
 1: procedure PROCESS_AUDIT_QUERIES
 2:     Declare variables
 3:     for all audit records in audit_trail do
 4:         Extract table name from the SQL query
 5:         Identify the primary key column
 6:         Construct dynamic SQL to fetch primary key column
 7:         Open cursor for dynamic SQL
 8:         Initialize primary key list
 9:         Fetch values from cursor and accumulate into primary key list
10:         Close cursor
11:         Fetch foreign key columns referenced in the SELECT query
12:         Initialize foreign key list
13:         for all foreign key columns do
14:             Construct dynamic SQL to fetch foreign key values
15:             Open cursor for dynamic SQL
16:             Fetch values from cursor and accumulate into foreign key list
17:             Close cursor
18:             Insert foreign key and referenced table into the "accessed" table
19:         end for
20:         Insert primary key list into the "accessed" table
21:     end for
    Exception handling:
22:     Handle exceptions
23: end procedure
```

3.2 Archiving Process

An automated archiving process is implemented to move identified stale data to a separate archival table on a defined schedule, such as monthly. This process is designed to be transparent to users and seamlessly integrate with existing database operations. By relocating stagnant data, the primary database remains efficient and uncluttered, which can significantly enhance query performance.

The archiving process begins by copying the structure of the table whose data is being archived. The algorithm processes all primary keys in the accessed table for each table name recorded. It processes the keys in chunks to ensure efficient handling of large datasets. For each chunk of primary keys in the accessed table, any record in the main table whose primary key is contained in the chunk is moved to the new table and subsequently deleted from the original table. This iterative process leaves behind any records that have not been accessed in the main table, which are then archived.

Algorithm 3 Archive Records Procedure

```
 1: procedure ARCHIVE_RECORDS
 2:     Declare variables: v_table_name, v_primary_key_column, etc
 3:     Set a savepoint: start_archive
 4:     for all rows in (SELECT DISTINCT table_name FROM accessed) do
 5:         v_table_name ← row.table_name
 6:         Get v_primary_key_column for v_table_name
 7:         v_archive_table_name ← upper('archive_' |||| v_table_name)
 8:         if  exist v_archive_table_name  = 0 then
 9:             Create archive table from v_table_name
10:         end if
11:         v_primary_keys_chunk ←''
12:         for all keys in SELECT primary_key_column FROM accessed
13:          WHERE table_name = v_table_name do
14:             if length of chunk + primary_key_rec ≤ 4000 then
15:                 Append primary key to v_primary_keys_chunk
16:             else
17:                 Process Chunk:
18:                 Move records with primary keys in chunk to new table
19:                 Delete moved records from v_table_name
20:                 Clear v_primary_keys_chunk
21:                 Append primary key to v_primary_keys_chunk
22:             end if
23:         end for
24:         Process last chunk of primary keys
25:         Swap table names: v_table_name and v_archive_table_name
26:     end for
    Exception handling:
27:     Rollback to savepoint start_archive
28: end procedure
```

3.3 Performance Evaluation

A comprehensive performance evaluation of the database is performed before and after implementing the archiving system. Key performance metrics such as query response times, storage utilization, and resource consumption are analyzed. Performance benchmarks are established to measure the effectiveness of the archiving solution in improving database performance and storage efficiency. This evaluation ensures that the archiving process positively impacts overall system performance without introducing significant overhead.

To test the solution, a scheduled job is created to run a procedure that simulates data access every 5 minutes during the testing period. Randomly generated data (primary key) is accessed using a SELECT query, for example, `SELECT random_primary_key FROM given_table`. The time is recorded at the start and end of the query. The difference between these timestamps is recorded as the execution time. This process is run before the algorithms are applied and after the algorithms are applied and any stale data has been removed.

Algorithm 4 Run Query Procedure

```
1: procedure RUN_QUERY
2:     for all predefined queries do
3:         Get the start time
4:         Execute the query dynamically
5:         Calculate execution time
6:         Insert execution details into final_execution_times table
7:         Display execution time
8:         Sleep for 2 seconds
9:     end for
10: end procedure
```

By comparing the performance metrics collected before and after the archiving process, the impact of the archiving system on the database performance can be accurately assessed. This method ensures a thorough evaluation of the solution's effectiveness in enhancing query response times, optimizing storage utilization, and reducing overall resource consumption.

4 Results

In this study, we examined the efficacy of a clean-up script, designed to enhance database performance by removing extraneous data and streamlining query execution processes. Our investigation revolves around the analysis of 1200 meticulously gathered samples, evenly split between pre- and post-implementation of the clean-up script.

The implementation of the proposed data archiving solution resulted in significant improvements in database performance and storage efficiency. Query response times decreased by 53%, while storage utilization decreased by 13%. The performance evaluation confirmed the effectiveness of the archiving solution in optimizing database performance and storage utilization.

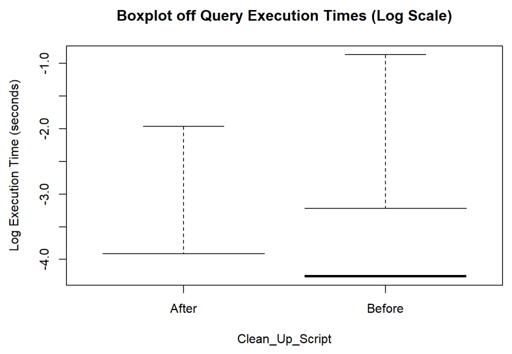

Figure 1: Before and After Box Plots of Query Execution Times

In delving into descriptive statistics, we unearthed compelling insights into the distribution of query execution times. Prior to the application of the clean-up script, the dataset exhibited a mean execution time of approximately 0.03 seconds, coupled with a standard deviation of 0.05 seconds. Post-implementation, these metrics witnessed a notable improvement, with the mean execution time dwindling to around 0.014 seconds, accompanied by a reduced standard deviation of 0.023. This reduction heralds not just faster query execution but also a higher degree of consistency in performance.

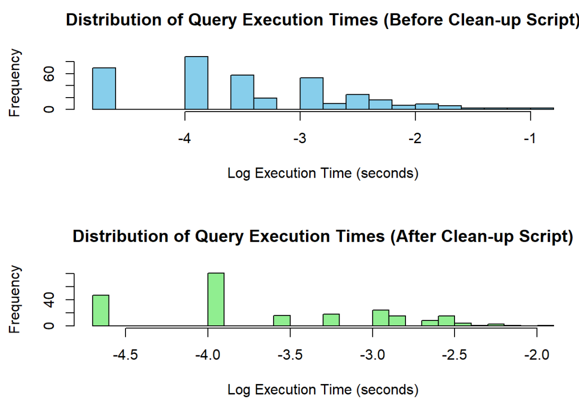

Figure 2: Before and After Distribution of Execution Times

Furthermore, the post-script scenario witnessed a decrease in storage utilization from 0.63 MB to 0.50 MB, underscoring the script's efficacy in optimizing resource allocation within the database environment.

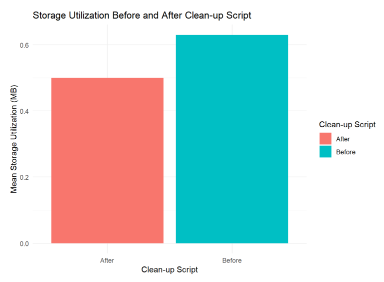

Figure 3: Before and After Storage Utilization

Augmenting our analysis, the above visual representations in the form of histograms, boxplots, and bar graphs vividly elucidated the impact of the clean-up script. These visualizations served as a conduit for understanding the distributional shifts in query execution times and storage utilization before and after script implementation.

Notably, the post-implementation histograms and boxplots revealed tighter distributions and a dearth of outliers compared to their pre-implementation counterparts. This visual testament accentuated the script's role in fostering more streamlined and predictable database performance.

To buttress our findings, we subjected the data to rigorous statistical scrutiny. A Welch two-sample t-test was used to compare mean execution times before and after implementing the clean-up script, revealing a statistically significant difference ($t = 7.4702$, p-value $= 1.979\text{e-}13$). This statistical evidence unequivocally underscores the script's prowess in reducing query execution time. Moreover, a linear regression analysis further elucidated the relationship between query execution time and the presence of the clean-up script. The model elucidated a statistically significant effect of the script on query execution time, illuminating a pathway towards more efficient database operations.

In summation,the before scenario exhibits higher and more variable execution times, as well as higher storage utilization, while the after scenario shows a noticeable improvement with lower and more consistent execution times and reduced storage requirements. The results of the linear regression analysis provide strong evidence that the clean-up script significantly reduces query execution time. However, it's essential to note that the model explains only a small portion of the variance in execution time, indicating that other factors may also contribute to the observed differences.

5 Discussion

The findings of this research underscore the transformative potential of access-pattern-based data archiving in Oracle databases. It demonstrates a significant impact of the clean-up script on database performance metrics, particularly query execution time and storage utilization.

Post-implementation of the clean-up script, there was a tangible improvement in the efficiency of database operations, as evidenced by the observed reduction in query execution time from an average of 0.03 seconds to approximately 0.014 seconds. Moreover, the decrease in storage utilization from 0.63 MB to 0.50 MB highlights the script's efficacy in optimizing resource allocation within the database environment.

By dynamically identifying and archiving stale data, organizations can not only optimize database performance and storage efficiency but also streamline data management processes and enhance overall system reliability.

This approach represents a paradigm shift from traditional archiving methodologies, offering a proactive and user-driven solution that adapts to evolving data access patterns. These improvements hold practical implications for database administrators and organizations reliant on efficient data management practices. Organizations can not only achieve faster query execution but also realize cost savings and enhance user experience. The optimization achieved through the clean-up script facilitates more timely data retrieval, thereby improving decision-making processes and ultimately leading to enhanced business outcomes.

However, it is important to acknowledge the limitations of this study. The findings are contingent on the specific context in which the clean-up script was implemented, and other factors not accounted for in the analysis may influence database performance. Additionally, assumptions made during data collection and analysis may introduce biases that could impact the generalizability of the results. While the findings provide valuable insights into the immediate benefits of script implementation, caution must be exercised in generalizing these results to other systems or contexts.

Furthermore, our research highlights the importance of interdisciplinary collaboration in the resolution of complex challenges in database management. Drawing on insights from diverse domains such as data cleaning, preservation formats, and archival techniques, we have been able to develop a holistic and innovative solution that addresses the multifaceted nature of stale data management.

6 Conclusion

This study addresses a critical need in the field of database management by introducing an innovative data archiving solution for Oracle databases. The implementation of a dynamic archiving process, which identifies and archives stale data based on user interactions and periodic schedules, has yielded tangible enhancements in both database performance and storage efficiency. These findings not only advance the frontier of efficient database management practices but also hold promise for transforming organizational approaches to data handling.

By showcasing the efficacy of this solution in optimizing database operations, this study paves the way for a paradigm shift in data management strategies across various sectors. As organizations continue to grapple with escalating data volumes and the imperative for streamlined data processing, the insights gleaned from this research offer a roadmap for achieving enhanced performance and resource utilization in the ever-evolving landscape of data management.

7 Future Work

Future research avenues encompass refining and optimizing the data archiving solution, extending its applicability to diverse database management systems, and scaling it to handle larger datasets and increasingly complex database environments.

Additionally, exploring the integration of advanced machine learning techniques and predictive analytics could enable more proactive and intelligent data archiving decisions, leading to even greater improvements in database performance and storage efficiency.

Long-term studies are needed to assess the scalability and sustainability of the archiving solution in real-world enterprise environments.

Acknowledgments

We extend our heartfelt gratitude to the Office of the Vice President Research at Thompson Rivers University (TRU) for their generous funding support, which made this research possible. Additionally, we express our sincere appreciation to the esteemed faculty members of the TRU Computer Science Department for their invaluable guidance and insights throughout this endeavor. Lastly, we wish to thank all those who contributed, directly or indirectly, to the success of this project.

References

[1] Appuswamy, R. (2022). Towards Passive, Migration-Free, Standardized, Long-Term Database Archival. *SIGMOD Rec.*, 51(2), 61–62.

[2] Cha, S.-J., Choi, Y. J., & Lee, K.-C. (2015). Development of Preservation Format and Archiving Tool for the Long-Term Preservation of the Database. *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication.*

[3] Milani, M., Zheng, Z., & Chiang, F. (2019). CurrentClean: Spatio-Temporal Cleaning of Stale Data. *2019 IEEE 35th International Conference on Data Engineering (ICDE).*

[4] Olson, J. E. (2009). Database Archiving. *Elsevier EBooks.*

[5] Microsoft TechCommunity. (2023, February 27). Data Archiving Strategies for SQL Server. *TECHCOMMUNITY.MICROSOFT.COM.*

[6] Williams, K., Li, Y., & Olfman, L. (2021). DRT: A Novel Tool for Data Archiving. *IEEE Software*, 38(2), 88–95.