Report for the Degree of Master of Computer Science

**Title of the report**

**Title: DRT-A Novel Tool for Data Archiving**



**Manish Shrestha (LC00015001879)**

**Masters in Computer Science (MCS) IIMS college**

**Computer Science Department**

**Lincoln University, Malaysia**

**June, 2024**

Research project for the Degree of Master of Computer Science

**Title of the report**

**Title: DRT-A Novel Tool for Data Archiving**

**Supervised by Prof. Subash Pandey**

A report submitted in partial fulfilment of the requirement for the degree of Master of Computer Science

**Manish Shrestha (LC00015001879)**

**Masters in Computer Science (MCS) IIMS college**

**Computer Science Department**

**Lincoln University, Malaysia**

**June, 2024**

Table of Contents

[**ABSTRACT** 5](#_Toc169447940)

[**1. INTRODUCTION** 1](#_Toc169447941)

[1.1 Background: 2](#_Toc169447942)

[1.2. Problem of Statement 3](#_Toc169447943)

[1.3. Research Questions 4](#_Toc169447944)

[1.5. Significance of Study 4](#_Toc169447945)

[Challenges with Exponential Data Growth: 5](#_Toc169447946)

[Diverse Strategies for Performance Enhancement: 5](#_Toc169447947)

[Building on DRT and NRRD Algorithm: 5](#_Toc169447948)

[Practical Application in Oracle Environment: 5](#_Toc169447949)

[Recognition of Limitation and Future Research: 5](#_Toc169447950)

[Choice of MySQL as an Alternative: 6](#_Toc169447951)

[1.6. Scope and Limitation of the Study 6](#_Toc169447952)

[**Scope:** 6](#_Toc169447953)

[**Limitations:** 7](#_Toc169447954)

[**2. LITERATURE REVIEW** 7](#_Toc169447955)

[2.1 Related theories 7](#_Toc169447956)

[2.2 Related works 8](#_Toc169447957)

[**3. RESEARCH METHODOLOGY** 11](#_Toc169447958)

[3.1 Research Framework/ Conceptual Framework/Theoretical Framework (Relationship between variables) 11](#_Toc169447959)

[3.2 Research tools 15](#_Toc169447960)

[3.3 Testing (Prototype) 16](#_Toc169447961)

[3.4 Data collection procedure: 17](#_Toc169447962)

[Data Collection Instruments: 18](#_Toc169447963)

[Data Analysis Methods: 18](#_Toc169447964)

[**4. EPILOGUE** 18](#_Toc169447965)

[4.1 Research timeline 18](#_Toc169447966)

[*Table 1: Research Timeline* 19](#_Toc169447967)

[4.2 Expected output 19](#_Toc169447968)

[*Table 2: 10% delete matrices* 21](#_Toc169447969)

[*Table 3: 50% delete matrices* 23](#_Toc169447970)

[**5. CONCLUSION** 27](#_Toc169447971)

[REFERENCE 28](#_Toc169447972)

**LIST of Tables**

[Table 1: Research Timeline 19](#_Toc169449025)

[Table 2: 10% Delete Matrices 21](#_Toc169449026)

[Table 3: 50% Delete Matrices 21](#_Toc169449027)

[Table 4: 90% Delete Matrices 22](#_Toc169449028)

**LIST of Figures**

[Figure 1: NRRD algorithm implementation 13](#_Toc169450735)

[Figure 2: 10% Delete 1 Index 24](#_Toc169450736)

[Figure 3: 10% Delete 10 Index 24](#_Toc169450737)

[Figure 4: 10% Delete 20 Index 24](#_Toc169450738)

[Figure 5: 50% Delete 1 Index 25](#_Toc169450739)

[Figure 6: 50% Delete 10 Index 25](#_Toc169450740)

[Figure 7: 50% Delete 20 Index 25](#_Toc169450741)

[Figure 8: 90% Delete 1 Index 26](#_Toc169450742)

[Figure 9: 90% Delete 10 Index 26](#_Toc169450743)

[Figure 10: 90% Delete 20 Index 26](#_Toc169450744)

# **ABSTRACT**

Effective database archiving is essential for managing data growth, enhancing system performance, and reducing storage costs. Structured Data Archiving (SDA) involves the safe and efficient removal of data from transactional processing databases (TPDs). Traditional methods of data deletion often require planned downtime or rely on SQL Delete commands, which can be slow and lead to issues such as locking and data fragmentation.

This paper builds on previous research by evaluating the Data Retirement Tool (DRT) implemented in Oracle, which utilizes the Near Replica Rapid Deletion (NRRD) algorithm. This algorithm expedites data removal by creating "near" replicas through partitioning exchange, showing superior performance over the standard SQL Delete method in terms of deletion speed and reduced system load.

In this study, we have extended the assessment of the NRRD algorithm by testing it within a MySQL environment. The results confirm that the NRRD algorithm retains its efficiency benefits in MySQL, outperforming traditional deletion methods. The findings suggest that the NRRD algorithm's efficiency improves with larger and more complex datasets. This research provides useful insights for database administrators and developers, demonstrating the algorithm's effectiveness across MySQL database management systems and its potential for enhancing data management practices in various operational contexts.

**Keywords:** Structured Data Archiving (SDA), Transactional processing databases (TPD), Near Replica Rapid Deletion (NRRD) algorithm, MySQL, PLSQL, Partitioning, Partition exchange.

# **1. INTRODUCTION**

Database plays a very vital role in every IT domain for proper/efficient storage and retrieval of data in a systematic manner. In the business world, databases enable the smooth operation of the organization by helping to store and organize huge volumes of data such that required data can be obtained whenever required for facilitating informed decision making. In healthcare, databases manage patient records, ensuring accurate and accessible medical histories. From finance to logistics, databases empower industries to harness the power of data, fostering innovation and productivity. In essence, the extensive use of databases is a testament to their indispensability across diverse sectors, contributing to the seamless functioning and advancement of various fields.

In the present context, the data are generated exponentially in every domain. As a result, the processing and retrieval of data from the database gets hampered. The increase in volume of data to be stored results in slower performance of the database operation. Accessing large volumes of data concludes undesirably higher cost operation. Therefore, proper management of a huge volume of data is a must. There are lots of ways in which the performance of databases can be enhanced. Some of which includes, Indexing [1], Partitioning, Caching [2], Query optimization [3], Load balancing [4], Database Sharding [5], Compression, and few more to mention. However, in this paper, discussion on how partitioning techniques can be used to enhance performance of a database which needs to organize and manage a huge volume of data in detail. In terms of partitioning technique for performance enhancement, several documents can be found where this technique has been used for data archiving as per the requirement. Such related works will be discussed in detail in the literature review section of this paper.

This paper will be based on a “DRT-A Novel Tool for Data Archiving” paper where an efficient technique of data archiving operation is discussed in detail. The main objective of the referenced paper was to enhance the database performance by using the data archiving technique. Furthermore, this article presents an innovative Data Retirement Tool (DRT) designed to eliminate data without necessitating system downtime or causing elevated system loads. Implemented within Oracle, the DRT features a Near Replica Rapid Deletion (NRRD) algorithm, allowing swift data removal from TPDs by generating "near" replicas through partitioning exchange. In comparison to the conventional SQL Delete method for data deletion, DRT exhibits superiority in terms of overall deletion time and minimized system load within the Oracle environment. The tool has been successfully licensed to a major airline for the removal of terabytes of data, showcasing substantial performance enhancements.

The referenced paper has presented the efficiency of DRT using the NRRD algorithm in an oracle environment only. I.e. for PL-SQL only. The referenced paper limitation includes future research requirements to investigate the efficiency of the NRRD algorithm in other RDBMS platforms and measure its performance. Therefore, MySQL will be used in this paper as an alternative to measure the performance efficiency of the NRRD algorithm.

## 1.1 Background:

The referenced paper explores the simulation of the Delete (DDL) statement using partitioning techniques and the partition exchange feature within Oracle's DBMS\_REDEFINITION package. While it showcases the efficacy of this approach, it lacks a detailed discussion on employing diverse partitioning techniques in various scenarios. This paper addresses the gap by delving into the use cases of different partitioning techniques, building upon the foundation of the referenced paper's data archiving technique.

The DBMS\_REDEFINITION package's partition exchange feature plays a crucial role, allowing seamless data exchange between partitions and non-partitioned tables. This refined method contrasts with traditional approaches, eliminating the need to drop or delete entire tables when executing data manipulations.

Our paper aims to provide practical examples of employing specific partitioning techniques, such as range, list, hash, and composite partitioning. Each technique offers unique advantages based on data characteristics and use cases. By extending the discussion beyond simulated DDL statements, this paper serves as a comprehensive guide for database administrators and developers, enhancing their understanding of database optimization and empowering them to make informed decisions in data management.

## 1.2. Problem of Statement

In the ever-evolving landscape of information technology, databases play a pivotal role in ensuring the efficient storage and retrieval of data across various domains. Whether in business, healthcare, finance, or logistics, databases serve as the backbone for organized data management, facilitating informed decision-making and contributing to the advancement of diverse fields. However, with the exponential growth of data, the processing and retrieval of information from databases face significant challenges, leading to a decline in performance.

The surge in data volume results in slower database operations, resulting in undesirable costs associated with accessing large datasets. To mitigate these challenges, various strategies such as indexing, caching, query optimization, and database sharding are employed. This paper focuses on the critical aspect of partitioning techniques as a means to enhance database performance, specifically when dealing with massive datasets that demand efficient organization and management.

The referenced paper introduces the innovative Data Retirement Tool (DRT), designed to streamline data elimination without causing system downtime or excessive system loads. Operating within the Oracle environment, the DRT employs a Near Replica Rapid Deletion (NRRD) algorithm, leveraging partitioning exchange to swiftly remove data from TPDs (Tables, Partitions, and Databases). The tool's superiority over conventional SQL Delete methods is evident in its reduced deletion time and minimized system load, as demonstrated in its successful implementation for a major airline, resulting in the removal of terabytes of data and substantial performance enhancements.

However, the referenced paper scope is only for the oracle database environment (PL-SQL). The paper has mentioned this as the future work / enhancement that needs to be addressed since there are other RDBMS available as well. Hence exploiting this future enhancement, this paper will use the foundation of the NRRD algorithm and apply it in MySQL and evaluate its performance accordingly.

## 1.3. Research Questions

This research seeks to delve into the details of partitioning as a performance enhancement strategy within the context of database management. Building upon the foundation laid by the "DRT-A Novel Tool for Data Archiving" paper, which highlights the efficiency of partitioning techniques, the focus is shifted to the application of this NRRD algorithm in MySQL RDBMS environment.

Specifically, the research will address the following sub-questions:

* Can the NRRD algorithm be applied efficiently in a MySQL RDBMS environment?
* Can the NRRD algorithm be applied to the MySQL RDBMS in the same way as it has been applied to the oracle RDBMS environment?
* Does the NRRD algorithm show its better efficiency against delete statements in the MySQL environment as it has done in the Oracle environment?

By answering these questions, this research aims to fill up the void that existed in the referenced paper. In this way, this paper will address one the future enhancement of the referenced paper.

## 1.5. Significance of Study

This research holds significant importance in the ever-evolving domain of database management, addressing critical challenges posed by the exponential growth of data in diverse domains. The study's significance are as follows:

### Challenges with Exponential Data Growth:

With data generation increasing exponentially, database operations face challenges in processing and retrieving data efficiently. The surge in data volume results in slower database performance and higher operational costs.

### Diverse Strategies for Performance Enhancement:

Numerous strategies exist to enhance database performance, including indexing, partitioning, caching, query optimization, and more. The study specifically delves into partitioning techniques and their role in optimizing database performance, with a focus on data archiving.

### Building on DRT and NRRD Algorithm:

The study builds upon the "DRT-A Novel Tool for Data Archiving" paper, which introduced the Data Retirement Tool (DRT) featuring the Near Replica Rapid Deletion (NRRD) algorithm. DRT, implemented within Oracle, demonstrated superior efficiency in data removal compared to traditional SQL Delete methods.

### Practical Application in Oracle Environment:

The referenced paper showcased the successful licensing of DRT to a major airline, resulting in the removal of terabytes of data and substantial performance enhancements. The efficiency of DRT using the NRRD algorithm, however, was limited to the Oracle environment, specifically PL-SQL.

### Recognition of Limitation and Future Research:

Acknowledging the limitation of the NRRD algorithm's applicability only to Oracle, the study recognizes the need for broader research to measure its efficiency in other RDBMS platforms.

### Choice of MySQL as an Alternative:

To address this limitation, the current study selects MySQL as an alternative RDBMS platform to measure the performance efficiency of the NRRD algorithm. This choice aims to ensure the generalizability of the algorithm's benefits across different database systems.

## 1.6. Scope and Limitation of the Study

### **Scope:**

The scope of the study revolves around the implementation and evaluation of the Data Retirement Tool (DRT) and its associated Near Replica Rapid Deletion (NRRD) algorithm. The key aspects within the scope include:

* **Database Archiving Strategies:** The study focuses on the broader context of database archiving strategies, with a specific emphasis on structured data archiving (SDA) and partitioning techniques.
* **Efficiency in Data Deletion:** The primary scope involves assessing the efficiency of DRT, which is designed to remove data from transactional processing databases (TPDs) without requiring system downtime and minimizing system load.
* **NRRD Algorithm Implementation:** The study delves into the implementation of the NRRD algorithm within DRT, exploring its functionality in creating "near" replicas through partitioning exchange for rapid data deletion.
* **Comparative Analysis with SQL Delete:** Comparative analysis is conducted between DRT's NRRD algorithm and the standard SQL Delete method, particularly focusing on overall deletion time and system load reduction.

### **Limitations:**

While the study provides valuable insights into data archiving and performance enhancement, it has certain limitations:

* **Oracle-Centric Implementation:** The study is limited to the implementation and evaluation of DRT and the NRRD algorithm within the Oracle environment (PL-SQL). The generalizability of findings to other database management systems may be constrained.
* **Specificity to Structured Data Archiving (SDA):** The focus is primarily on structured data archiving. The applicability and effectiveness of DRT and the NRRD algorithm for unstructured or semi-structured data scenarios are not explored.
* **Single Tool Evaluation:** The study concentrates on the evaluation of a single tool (DRT) and its associated algorithm. Alternative tools or methods for data archiving and deletion are not thoroughly explored or compared.
* **Dependency on Oracle Environment:** As the study explores the performance of DRT in Oracle, its adaptability and effectiveness in non-Oracle environments, including MySQL (as planned for future research), are left as areas for future investigation.

# **2. LITERATURE REVIEW**

## 2.1 Related theories

On the domain of optimization via partitioning technique, there are several papers available in which the respective authors have shown their expertise and domain knowledge for enhancing the performance of the database. Furthermore, multiple theories have been put forward by the authors in more or less similar environment and configuration. Some of the theories include:

* Data base optimization via partitioning and partitioning exchange with low system down time.
* Data archiving on the basis of various requirement into hierarchical storage. i.e. Online, near line, and offline.
* Automatic suggestion of partitioning technique on the basis of database configuration, number of table and the nature of table.

## 2.2 Related works

After the through search on the domain of database/query optimization via data archiving several related papers were referenced. Each paper was thoroughly read and on the basis of all the referenced papers the topic for this paper was chosen.

**DRT – A Novel Tool for Data Archiving** paper was written by Kevin Williams, Yan Li, and Lorne Olfman. The paper discussion on detail way of efficiently overcoming the time-consuming process of delete statement for a huge amount of data in a database. In order to solve this problem the authors have generated an algorithm named Near Replica Rapid Deletion (NRRD) for overcoming this issue and simulate the delete statement via partitioning technique. As per the metrics provided in this paper, the deployed algorithm has properly met the objective. i.e. the algorithm outperformed the delete operation in terms of execution time by taking less time to simulate the delete operation via partition technique. However the scope of this algorithm is limited up to PLSQL environment only. The efficiency of this algorithm is yet to be tested in other RDBMS. Hence exploiting this future enhancement work this project has been done in MySQL RDBMS. [6]

**Research and Application of Data Archiving based on Oracle Dual Database Structure** paper was written by Cui Jin, Naijia Liu and Li Qi. This research paper focuses on data archiving on Oracle Dual Databases structure. It has given more emphasis on importance of structured data in comparison to unstructured data. This paper discusses about data archiving technique on the basis of hierarchical storage. The data archiving process included four aspects namely: Analysis of business, analysis of storage, data analysis and technical analysis. The scope of this research paper includes highlighting the significance of data archiving in universities to support its day to day operation. On the other hand, the paper lacks on providing the detail discussion on the challenges and limitations that the authors had encountered during the structure data archiving based on the dual database structure of oracle. Furthermore, the focus on a specific database structure (Oracle Dual Database Structure) may restrict the generalizability of the findings to other database systems. [7]

**Reducing Data Access Time using Table Partitioning Techniques** paper was written by Veronika Šalgová, Karol Matiaško. This paper discusses in details about the importance of partitioning technique and its types. Furthermore, it provides more light on the performance of each partitioning technique and their performance in graphical and tabular format. The research focuses on the importance of fast data access due to the prevalence of large databases in various fields. It delves into partitioning techniques such as Hash, Range, and List, highlighting their benefits in improving performance, availability, and reducing costs. It discusses the implementation of partition keys and the significance of partition elimination for performance improvement in querying large tables. The study primarily focuses on the benefits and comparisons of data access times with and without partitions, potentially limiting the exploration of other factors influencing data access efficiency. The research may not extensively cover all possible partitioning techniques or scenarios, potentially limiting the generalizability of the findings to all database systems. The scope of the study may not address specific challenges or constraints related to the practical implementation of partitioning techniques in real-world database systems. [8]

**Automated Partitioning Design in Parallel Database Systems** paper was written by Rimma Nehme, and Nicolas Bruno. This paper introduces a partitioning advisor that recommends optimal partitioning designs based on expected workloads. The tool suggests which tables should be replicated across compute nodes and which ones should be distributed according to specific columns, aiming to minimize the cost of similar workloads. Notably, the proposed techniques are deeply integrated with the parallel query optimizer, leading to more accurate recommendations in a shorter time. Experimental evaluation on a real MPP system, Microsoft SQL Server 2008 Parallel Data Warehouse, with both real and synthetic workloads demonstrates the effectiveness of the techniques and underscores the importance of seamless integration between the partitioning advisor and the underlying query optimizer. The scope of the research paper focuses on designing a partitioning advisor for parallel database systems to recommend the best partitioning design for an expected workload. It discusses the integration of partitioning techniques with the parallel query optimizer to provide accurate recommendations efficiently. The paper delves into the disadvantages of shallowly-integrated approaches in partitioning configurations due to the large search space and expensive evaluation processes, proposing a deeper integration with the query optimizer as a solution. It highlights the impact of replication bounds on partition recommendations, showing that certain replication bounds can improve the quality of recommendations, but excessive replication may not lead to smaller cost configurations .The paper does not delve into the details of parallel optimization, as it is outside the paper's scope. [9]

**Efficient Bulk Deletes in Relational Databases** The paper by A. Gärtner, A. Kemper, D. Kossmann, and B. Zeller introduces a novel vertical approach to optimize DELETE operations in relational databases, particularly within B+-tree indices. Traditional row-by-row deletions are inefficient due to random disk I/O. Their new method enhances performance by first removing records from the base table, then from indices. A key innovation is the bulk delete operator (↓), which identifies and deletes records before reorganizing the table or index. The paper evaluates strategies like sorting and merging, hashing, and range partitioning, with the query optimizer selecting the best approach based on workload characteristics. It also addresses clustered and unique indices, concurrency control measures, and recovery techniques like checkpoints. The vertical method proves significantly more efficient, offering a faster and more effective solution for database systems. [10]

**An Efficient Method for Performing Record Deletions and Updates Using Index Scans** paper addresses inefficiencies in traditional DBMS methods for record deletions and updates. These inefficiencies stem from a lack of synergy between query planning and data manager components, leading to unnecessary lock calls and multiple index traversals. This issue is exacerbated when range scans span multiple leaf pages, causing significant performance degradation. The paper emphasizes the need for RDS (Relational Data System) designers to understand data manager processing deeply to improve performance and optimize cost models. Research has explored improving coordination between query planners and data managers. Enhancements in index maintenance algorithms, for example, have shown potential in reducing overheads related to lock management and page traversals. Aligning the query planner's expectations with the data manager's capabilities can lead to more efficient execution plans. The implementation of this method in DB2 V7 showcases its practical benefits, addressing specific customer requirements and improving benchmark performances, such as the TPC-H benchmark. This real-world benchmark reflects typical database workloads, and improvements here can significantly enhance query execution times and overall system throughput. [11]

# **3. RESEARCH METHODOLOGY**

## 3.1 Research Framework/ Conceptual Framework/Theoretical Framework (Relationship between variables)

**Research Framework:** The research framework in this context revolves around the development, implementation, and evaluation of a database archiving strategy, particularly the introduction of the Data Retirement Tool (DRT) and its associated Near Replica Rapid Deletion (NRRD) algorithm. The primary objective is to address challenges in data archiving, including issues related to system downtime, slow performance during the deletion process using MySQL RDBMS environment.

**Conceptual Framework:**

* **Database Archiving Strategies:** The conceptual framework centers on the concept of database archiving as a strategy for organizations to control data growth, enhance system performance, and reduce storage costs.
* **Structured Data Archiving (SDA):** Within the conceptual framework, Structured Data Archiving (SDA) is identified as a key component, emphasizing the need to safely and accurately remove data from transactional processing databases (TPDs).
* **Challenges with Existing Approaches:** The framework acknowledges existing challenges associated with data deletion methods, such as planned database downtime and the use of SQL Delete commands, which result in performance issues like slowness, locking, and data fragmentation.
* **Introduction of Data Retirement Tool (DRT):** The introduction of the Data Retirement Tool (DRT) is a pivotal concept, representing an innovative solution designed to remove data without system downtime and minimize increased system load.
* **Near Replica Rapid Deletion (NRRD) Algorithm:** The conceptual framework incorporates the Near Replica Rapid Deletion (NRRD) algorithm as a theoretical concept within the DRT, focusing on its role in rapidly removing data from TPDs by creating "near" replicas through partitioning exchange.
* **Performance Comparison with SQL Delete:** The framework involves a comparison between the DRT and the standard SQL Delete method, evaluating overall deletion time and system load reduction as key performance indicators.

**Theoretical Framework (Relationship between Variables):**

The theoretical framework establishes relationships between key variables:

* **Independent Variable:** Implementation of the Data Retirement Tool (DRT) with the NRRD algorithm.
* **Dependent Variables:** Overall deletion time and system load reduction in MySQL.

The framework posits that the introduction of the DRT with the NRRD algorithm as an independent variable will lead to improvements in the dependent variables—reduced overall deletion time and decreased system load in MySQL, as compared to the standard SQL Delete method.

In summary, the research framework revolves around the conceptualization of database archiving strategies, the introduction of the Data Retirement Tool (DRT), and the theoretical relationships between variables such as the NRRD algorithm, deletion time, and system load reduction in the MySQL environment.

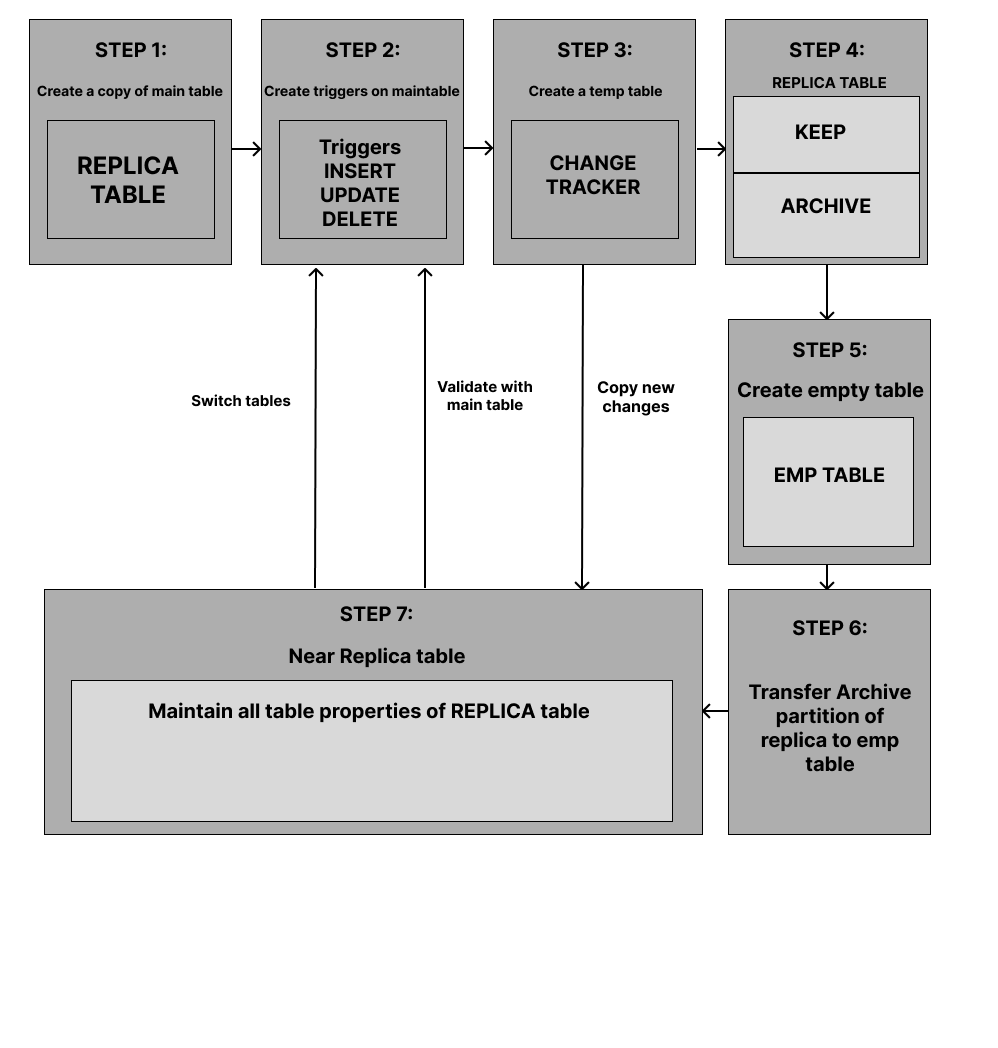


Figure 1: NRRD algorithm implementation

**NRRD algorithm implementation overview:**

First of all a table is selected in which data archive needs to be done. Let us consider this table as an original/main table.

* **Step 1: Create a replica of original table**
  + The replica/exact copy of the selected original table is made. Let the table be named as Replica table.
* **Step 2: Creation of triggers**
  + Trigger are created in the main table such that every changes that are made in the main table can be logged into a log table.
* **Step 3: Creation of a log table**
  + A log table is specially designed for maintaining or storing the changes that are made in the main table during the processing of this algorithm in order to maintain the concurrency. Let the table be named as change tracker.
* **Step 4: Partitioning replica table**
  + The replica table is modified to have to partition. ‘Archive’ and ‘Keep’ partition. ‘Archive’ partition is meant to store all the records of table in which the flag is archive. Similarly, the ‘Keep’ partition is meant to store all the records having ‘Keep’ partition.
* **Step 5: Creation of an empty table**
  + Create an empty table named ‘Emp\_table’. It is created to store only archive date.
* **Step 6: Partition exchange or transfer**
  + By using the partition exchange technique, transfer the archive partition to the Emp\_table. (Emp\_table has no partition)
* **Step 7: Synchronization with original table**
  + During the execution of the algorithm, there might me some change done in the main table. Hence, in order to maintain the integrity of data and to reflect all the necessary change to the replica table, we need to check and make changes in replica table accordingly to make replica table an exact copy of main table.
* **Step 8: Future data management**
  + When the necessary change fail to be reflected in the replica table then the whole process needs to be repeated. On the other if all the changes are reflected then the replica table is renamed to main table in which all the records would have keep flag only. I.e. Archive partition is empty. In future when the table start to get populated again with new data then such data are aligned to keep and archive partition on the basis of their flag.

## 3.2 Research tools

The referenced paper provides a detailed discussion on applying the NRRD algorithm to simulate delete statements using partitioning techniques and partition exchange. In this current study, we enhance the original work by implementing the NRRD algorithm in a MySQL RDBMS environment, as the previous research focused exclusively on an Oracle environment. This enhancement aims to showcase the algorithm's adaptability and effectiveness in MySQL, thereby extending its applicability and providing insights into its performance across different database management systems.

**Resource utilization specification:**

**In referenced paper hard specification:**

* **Database Software:** ORACLE Enterprise Edition version 11.2.0.4
* **Operating System:** Red Hat Enterprise Linux 6
* **Memory:** 128 GB
* **Processor:**
  + **Type:** Single CPD
  + **Cores:** 12 cores

**In this paper hard specification:**

* **Database Software:** MySQL Community Server (8.0.36)
  + **Tool:** MySQL Workbench
* **Host operating System**: Windows 11 (64-bit operating system, x64-based processor)
* **Installed RAM** : 8 GB
* **Processor:** AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz
* **Storage Capacity:** 227 GB

The referenced paper's experimental setup is designed for high-performance and intensive database operations with a focus on maximizing memory and storage throughput. In contrast, the current paper's hardware configuration is more typical of a mid-range computing environment, balancing performance and cost-efficiency. This disparity in resource utilization underscores the difference in computational capabilities and potential performance outcomes between the two setups.

## 3.3 Testing (Prototype)

In the context of using MySQL for applying the Data Retirement Tool (DRT), the testing of the prototype involves the following considerations:

**Prototype development:** The NRRD algorithm has already been implemented in a PLSQL environment, as documented in the referenced paper. However, testing the algorithm's functionality and efficiency in other relational database management systems (RDBMS) has not been conducted yet. Consequently, exploring the algorithm's performance across different RDBMS environments has been identified as a potential area for future work in the referenced paper. Building on the foundation laid by the referenced paper, this current study focuses on implementing the NRRD algorithm in MySQL. The goal is to evaluate the efficiency and effectiveness of the DRT tool when deployed in a MySQL environment. By doing so, this paper aims to provide insights into the adaptability and performance of the NRRD algorithm beyond its original PLSQL implementation, thereby broadening its applicability and utility across different database systems.

**Testing objective:** The primary goal of this paper is to thoroughly evaluate the performance and reliability of the NRRD algorithm within a MySQL environment. This objective will be achieved by conducting a comprehensive series of tests designed to determine whether the algorithm functions as intended under different test case scenario. The test cases that has been used in referenced paper has been used in this paper as well to evaluate the performance. The test cases are as follows:

**Test cases:**

* **Table size:** ~1.5 million records
* **Percentage of rows deleted:** 10%, 50% and 90%
* **Number of indices:** small (1), medium (10), and large (20)

The "percentage of rows deleted" is employed to examine the diverse performance characteristics that may arise when varying amounts of data are deleted from a table. Deleting a row involves more than just eliminating the row itself; it also necessitates removing its references from any associated indices, which can be a resource-intensive process. Additionally, this operation requires rebalancing the index trees to maintain the integrity and efficiency of the database's indexing system. As a result, the quantity and complexity of indices associated with a table can significantly influence the overall performance and duration of the deletion process. The more indices a table has, the more work the database system must perform to update these structures, leading to potentially greater computational overhead and longer processing times. Therefore, understanding the impact of varying deletion percentages helps in optimizing database performance, particularly in scenarios involving large-scale data modifications.

## 3.4 Data collection procedure:

For the purpose of testing and evaluating the Data Retirement Tool (DRT) prototype in the MySQL environment, a set of dummy data has been generated via SQL script. The script is designed to simulate realistic scenarios and workloads, allowing for a comprehensive assessment of the tool's performance under varying conditions. In this paper, dummy data of an airline has been generated. Furthermore, in the referenced paper also airline related dataset has been used. However, both the dataset are not same. In order to evaluate the efficiency of the NRRD algorithm in MySQL, the volume of data is more important hence, the dummy dataset was generated using script.

### Data Collection Instruments:

The primary instrument for data collection involves the SQL script designed to generate dummy data within the MySQL database. This script creates a representative dataset that mirrors real-world conditions, ensuring that the DRT's functionality is tested against diverse scenarios. As the data is generated through the SQL script, the sampling process involves the comprehensive execution of queries within MySQL to cover a range of scenarios. This includes scenarios with varying data sizes, complexities, and workload patterns.

­

### Data Analysis Methods:

The analysis of the collected data revolves around interpreting the results obtained from the execution of the SQL script. This involves assessing the DRT's performance in terms of deletion time, system load within MySQL, and other MySQL-specific parameters. By utilizing dummy data generated through the SQL script, the data collection procedure aims to provide a controlled yet realistic environment for evaluating the DRT prototype's effectiveness and efficiency in a MySQL setting. This approach ensures that the tool's performance is assessed under conditions that mimic actual usage scenarios.

## **4. EPILOGUE**

## 4.1 Research timeline

|  |  |  |
| --- | --- | --- |
| **Activity** | **Time Frame** | **Remarks** |
| Project Initiation and Planning | 2023-12-15 to 2024-01-15 | Project idea generation and findings on the selected topic. |
| Data Collection | 2024-01-15 to 2024-01-18 | Data generation |
| Prototype Development and Implementation | 2024-01-18 to 2024-03-20 | Application of the algorithm |
| Testing | 2024-03-20 to 2024-03-25 | Testing the result against the desired objective. |
| Data Analysis and Interpretation | 2024-03-25 to 2024-03-31 | Interpreting the result. |
| Report Writing and Documentation | 2023-12-15 to 2024-03-31 | Documenting each activity, findings and process of the project. |

Table 1: Research Timeline

## 4.2 Expected output

As per the output obtained from referenced paper, the result were positive in nature. The deletion of huge volume of data took significant amount of time whereas the NRRD algorithm was able to simulate the delete operation using partition technique and perform the operation in very less time compared to traditional way of bulk delete. Furthermore, the output also depends on the number of index the table has. The higher the number of index, the higher is the execution time of deletion operation and vice versa.

However, after implementing the algorithm in MySQL environment, there were some test cases where the result were negative and for rest of the test cases the result were positive. Furthermore, in the test case environment where the test cases were conducted, ~1.5 million rows were considered. The cases where, there was 1 index & less percent of data to be deleted, the NRRD algorithm could not outperform the deletion operation. On the other hand, on the test cases where there were more number of index and more percent of data deletion was required the NRRD algorithm outperformed the deletion operation.

In last step, thus obtained replica table needs to be tested if it is in sync with original table or not. There might be some changes done in the original table during the operation of NRRD algorithm. In this way while maintaining the concurrency there might be the risk of not maintaining the data integrity between original table and replica table. Hence the changes made in original needs to be reflected in replica table. However, the referenced paper has not mentioned anything about the quantity of rows update done in the original table as millions of row update can take significant amount of time. Furthermore, in this case there occur the issue of number of locks insufficient. Therefore, in this way this paper has provided the insights for the efficiency of NRRD algorithm in MySQL environment. Following are the results depicted in tables and graphs:

**List of tables:**

**10% Delete**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **10% delete 1 Index** | | **10% delete 10 Index** | | **10% delete 20 Index** | |
| **Without partition** | **With partition** | **Without partition** | **With partition** | **Without partition** | **With partition** |
| 8.064 | 66.939 | 39.892 | 66.181 | 83.391 | 68.000 |
| 7.890 | 67.352 | 37.375 | 67.068 | 83.641 | 68.194 |
| 8.125 | 66.641 | 38.282 | 69.115 | 83.466 | 68.501 |
| 7.643 | 66.769 | 38.121 | 68.766 | 82.969 | 68.632 |
| 7.763 | 66.135 | 38.545 | 66.689 | 82.177 | 68.387 |
| 8.606 | 67.305 | 39.198 | 68.199 | 83.126 | 68.578 |
| 8.725 | 66.007 | 37.267 | 67.830 | 82.140 | 68.417 |
| 8.635 | 67.163 | 37.689 | 66.612 | 82.092 | 68.602 |
| 8.715 | 67.951 | 39.075 | 68.165 | 82.101 | 68.356 |
| 7.702 | 67.758 | 38.184 | 68.823 | 83.948 | 68.698 |
| 8.704 | 66.883 | 39.603 | 66.678 | 83.215 | 68.503 |
| 8.651 | 66.275 | 39.156 | 66.210 | 83.985 | 68.579 |
| 8.621 | 67.125 | 39.245 | 67.789 | 82.970 | 68.432 |
| 7.693 | 68.386 | 38.678 | 67.666 | 82.079 | 68.654 |
| 7.645 | 66.123 | 37.112 | 68.190 | 82.212 | 68.408 |
| 8.667 | 66.075 | 38.193 | 67.815 | 82.846 | 68.521 |
| 7.708 | 67.228 | 37.635 | 66.102 | 83.838 | 68.231 |
| 8.670 | 67.276 | 38.178 | 66.626 | 83.937 | 68.912 |

Table 2: 10% Delete Matrices

**50% delete**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **50% delete 1 Index** | | **50% delete 10 Index** | | **50% delete 20 Index** | |
| **Without partition** | **With partition** | **Without partition** | **With partition** | **Without partition** | **With partition** |
| 23.453 | 66.970 | 223.188 | 68.374 | 311.823 | 67.564 |
| 23.422 | 68.982 | 221.188 | 66.592 | 318.344 | 67.847 |
| 22.169 | 66.059 | 223.142 | 68.938 | 308.546 | 67.150 |
| 22.278 | 66.529 | 222.456 | 65.589 | 310.978 | 67.289 |
| 23.482 | 67.613 | 225.312 | 66.965 | 308.154 | 67.467 |
| 22.096 | 66.197 | 225.314 | 68.356 | 303.634 | 67.954 |
| 22.351 | 66.286 | 226.859 | 67.721 | 310.382 | 67.358 |
| 23.238 | 66.179 | 222.507 | 67.768 | 306.975 | 68.134 |
| 22.414 | 68.912 | 223.984 | 68.389 | 315.218 | 67.415 |
| 22.337 | 68.871 | 223.465 | 68.566 | 310.521 | 68.088 |
| 22.110 | 66.478 | 225.999 | 66.836 | 311.847 | 67.291 |
| 23.273 | 67.308 | 224.745 | 66.675 | 311.789 | 68.201 |
| 22.455 | 66.332 | 223.876 | 67.692 | 318.654 | 68.243 |
| 22.179 | 66.543 | 225.534 | 68.656 | 315.362 | 68.062 |
| 22.292 | 68.165 | 226.783 | 68.509 | 314.372 | 67.218 |
| 23.407 | 67.107 | 221.412 | 66.635 | 313.413 | 67.278 |
| 22.067 | 67.894 | 224.398 | 68.801 | 311.982 | 68.376 |
| 22.359 | 68.620 | 222.761 | 67.169 | 318.201 | 67.993 |

Table 3: 50% Delete Matrices

**90% Delete**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **90% delete 1 Index** | | **90% delete 10 Index** | | **90% delete 20 Index** | |
| **Without partition** | **With partition** | **Without partition** | **With partition** | **Without partition** | **With partition** |
| 37.267 | 67.181 | 396.750 | 69.812 | 573.282 | 69.545663 |
| 38.969 | 69.673 | 397.281 | 67.970 | 575.234 | 68.387322 |
| 37.656 | 66.819 | 397.220 | 66.555 | 572.516 | 68.438921 |
| 37.189 | 67.201 | 396.765 | 68.933 | 575.391 | 68.494428 |
| 38.945 | 69.732 | 397.312 | 67.169 | 573.245 | 69.572914 |
| 37.604 | 66.853 | 397.201 | 67.285 | 574.789 | 68.401275 |
| 37.298 | 67.168 | 396.798 | 69.905 | 572.621 | 68.447398 |
| 38.978 | 69.659 | 397.374 | 67.882 | 575.453 | 68.479521 |
| 37.672 | 66.799 | 397.265 | 66.578 | 573.901 | 69.559832 |
| 37.256 | 67.246 | 396.721 | 68.950 | 574.365 | 68.410986 |
| 38.935 | 69.683 | 397.298 | 67.142 | 572.748 | 68.426512 |
| 37.681 | 66.775 | 397.184 | 69.843 | 575.679 | 68.486779 |
| 37.224 | 67.134 | 396.786 | 68.024 | 573.514 | 69.537286 |
| 38.952 | 69.701 | 397.329 | 66.534 | 574.092 | 68.398735 |
| 37.645 | 66.824 | 397.239 | 68.975 | 572.384 | 68.444289 |
| 37.283 | 67.219 | 396.774 | 67.195 | 575.932 | 68.492105 |
| 38.961 | 69.648 | 397.288 | 69.765 | 573.687 | 69.528961 |
| 37.612 | 66.781 | 397.201 | 68.010 | 574.528 | 68.389654 |

Table 4: 90% Delete Matrices

**List of figures**

Figure 2: 10% Delete 1 Index

*Figure 3: 10% Delete 10 Index*

Figure 4: 10% Delete 20 Index

Figure 5: 50% Delete 1 Index

Figure 6: 50% Delete 10 Index

Figure 7: 50% Delete 20 Index

Figure 8: 90% Delete 1 Index

Figure 9: 90% Delete 10 Index

Figure 10: 90% Delete 20 Index

## **5. CONCLUSION**

After the implementation of NRRD algorithm in MySQL platform, several test cases were carried out. These test cases are the same test cases that had been carried out in the referenced paper.

As per the carried put test cases, there were positive results in some cases and negative results in some cases. The positive results indicates that, the NRRD algorithm outperformed the traditional way of deleting huge volume of records. On the other hand, the negative results indicates the traditional way of deleting huge volume of records outperformed the NRRD algorithm.

In this paper, around ~1.5 million rows of data are used whereas in the referenced paper 40 million rows are used. After analyzing the gathered metrics, it can be seen that, when the volume of data to be deleted or archived increases along with the complexity of table then the NRRD algorithm efficiency increases. And in contrast, when the volume of data to be deleted or archived decreases along with the complexity of table then the efficiency of NRRD algorithm is same or lower than the traditional way of deleting rows. During the experiment, when the number of indexes and volume of data to be deleted increases then the performance of NRRD algorithm is better than traditional method of row deletion and vice versa.

Hence, even though some of the test cases are negative, considering the metrics from this experiment conducted, it can be concluded that with the increase in data that needs to be archived and increase in complexity of table the NRRD algorithm outperforms the traditional method of deleting data row by row in terms of time and efficiency.

## 5.1 Key Findings

The implementation of the Near Replica Rapid Deletion (NRRD) algorithm in a MySQL environment has been comprehensively examined in this study. The research aimed to evaluate the performance and scope of the NRRD algorithm in comparison to traditional data deletion methods. The primary findings from this implementation are detailed below:

**Performance Comparison:** The NRRD algorithm showed better performance than traditional data deletion methods in certain test cases. However, there were situations where the traditional method did not outperformed the NRRD algorithm, indicating that the effectiveness of a deletion strategy is highly dependent on context and data characteristics.

**Impact of Data Volume and Indexes:** Detailed metric analysis indicated that the efficiency of the NRRD algorithm significantly improves with increased data volume and the number of indexes in the table. Specifically, as the dataset size and table complexity increase, the NRRD algorithm consistently outperformed the traditional row deletion method. This trend suggests that the benefits of the NRRD algorithm become more evident with larger datasets.

**Execution Time Dynamics:** The study found that the time difference between the NRRD algorithm and the traditional deletion method increases as the volume of data and the number of indexes increase. This is due to the increased workload on the database system, where more indexes require extensive updates during row deletions. As a result, the delete operation takes considerably longer with large data volumes and numerous indexes. This finding highlights the importance of managing indexes carefully; while they enhance database performance, excessive use can hinder deletion operations.

**Dataset Size Comparison:** This study used approximately 1.5 million rows of data for testing, while the referenced paper used about 40 million rows. Despite the smaller dataset in this study, the metrics indicate that the NRRD algorithm's performance improves with larger datasets. It can be concluded that, although some test cases yielded negative results, the NRRD algorithm is likely to deliver positive outcomes across all test cases with an increased volume of data, as demonstrated in the referenced paper. This suggests that the algorithm's efficiency scales well with data size.

**Implications for Database Management:** The findings highlight the importance of considering data volume and the number of indexes when implementing the NRRD algorithm. The results advocate for a balanced approach to indexing, recommending its use primarily when necessary rather than solely for performance enhancement. This approach ensures that indexes contribute to overall database efficiency without negatively affecting deletion performance.

In summary, this study confirms that the NRRD algorithm, despite some initial limitations, shows promising results for large-scale data deletion in MySQL environments. Future research should investigate the algorithm’s performance with even larger datasets and across various database systems to further validate its effectiveness and robustness.

## 5.2 Recommendations

The effectiveness of NRRD algorithm should be explored in different types of RDBMS. In this paper only MySQL has been used to check the efficiency of the NRRD algorithm.

The paper should have provided the information about the nature of dataset being used. Information known about the dataset was, it was a dataset of an airline. There is not much information about the dataset which was used during the algorithm implementation.

The referenced paper has gone through many papers and researched a lot about the partitioning strategy. However, the paper has not discussed about the partitioning technique that should be used for the NRRD algorithm. This aspect is very important because, on the basis of nature of dataset, different types of partitioning technique must be applied. Different types of partitioning technique have their own unique characteristics and based on the characteristic of dataset, suitable partitioning technique must be applied for best and optimal performance.

Furthermore, the referenced paper has discussed about the synchronization of replica table with respect to original table. All the change done in original table must be reflected on the replica table. However, the paper has not discussed about the amount of changes to be done and its synchronization with the original table. If huge volume of data needs to be changed or updated then it may take significant amount of time to do so. This may hamper the matrices that was gather. In contrast, if the amount of change which needs to be done is not huge in volume then it may not hamper the gathered matrices.

The referenced paper has done significant amount of study regarding the lock of the database system. However, the paper has not discussed about the lock of the database system and the problem that may arise when dealing with huge volume of data. The common issue would be insufficient number of locks. This issue arises when dealing with huge volume of data without the partition. As a result of this gathering matrices was difficult when huge volume of data needed to be archived and when the table complexity was very high.

Gathering matrices was difficult, tedious and time consuming as it had to be rechecked multiple time for its reliability. In some cases due to cache memory, the obtained matrices would have less amount of execution time. Hence under such scenario, tables had to be flushed, log file had to be deleted, memory needed to be cleared and sometimes server needed to be restarted. Information regarding such issue has not been provided in the referenced paper.

MySQL time schedular can be used to automate the task of archiving.

# REFERENCE

|  |  |
| --- | --- |
| [1] | H. G. A. L. M. S. A. T. M. F. Siti Maesaroh, "Query optimization in MySQL Database Using Index," *International Journal of Cyber and IT Services Management,* 2022. |
| [2] | N. B. a. H. A. Mantu Kumar, "Cache Based Query Optimization Approach in Distributed Database," *International journal of Computer Science Issue,* vol. Vol.9, no. Issue 6, 2012. |
| [3] | J. K. Matthias Jarke, "Query optimization in Database System," *ACM Computing survey.* |
| [4] | D. F. P. V. Luc Bouganim, "Dynamic Load Balancing in Hierarchical Parallel," *INRIA.* |
| [5] | B. M. Abdelhafiz, "Distributed Database Using Sharding Database Architecture," *IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE). IEEE,* 2020. |
| [6] | Y. L. a. L. O. K. Williams, "DRT: A novel tool for data archiving," *IEEE Softw,* vol. vol. 38, March 2021. |
| [7] | N. L. a. L. Q. C. Jin, "Research and application of data archiving based on oracle dual database structure," *Journal of Software,* vol. vol. 7, 2012. |
| [8] | V. S. a. K. Matiasko, "Reducing Data Access Time using Table Partitioning Techniques," *ICETA 2020 - 18th IEEE International Conference on Emerging eLearning Technologies,* Nov 2020. |
| [9] | T. Sellis, "Automated Partitioning Design in Parallel Database Systems," *ACM Digital Library.,,* 2011. |
| [10] | A. K. D. K. a. B. Z. A. Gärtner, "Efficient Bulk Deletes in Relational Databases.," *Proceedings 17th International Conference on Data Engineering. IEEE, 2001,* 2001. |
| [11] | W. C. S. S. S. B. R. P. S. Manoj Poudel, "An Efficient Method for Performing Record Deletions and Updates Using Index Scans," *Proceedings of the 28th Internaltional Conference on Very Large Databases.,* 2002. |