International Journal of Innovative Engineering Applications

Journal homepage: https://dergipark.org.tr/ijiea

OPTIMIZING BIG DATA MANAGEMENT ON MICROSOFT SQL SERVER: ENHANCING PERFORMANCE THROUGH NORMALIZATION AND ADVANCED ANALYTICAL TECHNIQUES

Süleyman Burak Altınışık[1], *Turgay Tugay Bilgin*[*2]

[1]*Department of Project Management, Mert Software Electronics, Bursa, Turkey*
[2]*Department of Computer Engineering, Faculty of Enginerring and Natural Sciences, Bursa Technical University, Bursa, Turkey*

Abstract

Original scientific paper

This study investigates Big Data management challenges and solutions in cable manufacturing using Microsoft SQL Server (MSSQL), focusing on performance optimization, normalization, and advanced analytical techniques. Addressing the 4Vs of Big Data, our case study collects data from 45 TAGs at one-minute intervals, generating approximately 56 million daily records. We employ OPC technology for data acquisition, strategic normalization processes, and advanced MSSQL optimization techniques. Normalization significantly reduced data redundancy, decreasing the dataset from 56 million to 283 rows per day and improving query execution times from over 40 minutes to less than 0.1 seconds for complex analytical queries. We also propose a database-independent software development approach to balance cost and performance. This research contributes practical insights into performance optimization, scalability, and cost-effective solutions for organizations managing large-scale data processing challenges in industrial settings, offering a blueprint for efficient Big Data management that balances technical performance with economic considerations.

Keywords: *Big data management, data normalization, cable manufacturing, performance optimization.*

MİCROSOFT SQL SERVER' DA BÜYÜK VERİ YÖNETİMİNİN OPTİMİZASYONU: NORMALİZASYON VE İLERİ ANALİTİK TEKNİKLER İLE PERFORMANSININ ARTIRILMASI

Özet

Orijinal bilimsel makale

Bu çalışma, kablo üretim sektöründe Microsoft SQL Server (MSSQL) kullanarak Büyük Veri yönetimi zorluklarını ve çözümlerini incelemekte, performans optimizasyonu, normalizasyon ve ileri düzey analitik tekniklere odaklanmaktadır. Büyük Veri'nin 4V'sini ele alan örnek olay incelemesinde, 45 TAG'dan bir dakikalık aralıklarla veri toplanmakta ve günlük yaklaşık 56 milyon kayıt oluşturulmaktadır. Veri toplamak için OPC teknolojisini, stratejik normalizasyon süreçlerini ve ileri düzey MSSQL optimizasyon tekniklerini kullanmaktayız. Normalizasyon, veri tekrarını önemli ölçüde azaltmış, veri setini günde 56 milyondan 283 satıra düşürmüş ve karmaşık analitik sorgular için sorgu yürütme sürelerini 40 dakikadan 0.1 saniyenin altına indirmiştir. Ayrıca, maliyet ve performans dengesini sağlamak için veritabanından bağımsız yazılım geliştirme yaklaşımı önermekteyiz. Bu araştırma, endüstriyel ortamlarda büyük ölçekli veri işleme zorluklarıyla karşılaşan organizasyonlar için performans optimizasyonu, ölçeklenebilirlik ve maliyet etkin çözümler konusunda pratik bilgiler sunmakta, teknik performans ile ekonomik hususlar arasında denge kuran etkili bir Büyük Veri yönetimi için bir yol haritası sunmaktadır.

Anahtar Kelimeler: *Büyük veri yönetimi, veri normalizasyonu, kablo üretimi, performans optimizasyonu.*

1 Introduction

The rapid evolution of Big Data has transformed data management and analytics, creating new opportunities and challenges for organizations across industries and academia [1]. With data volumes growing exponentially, effectively managing, processing, and analyzing large datasets has become a crucial skill [2]. This growth is characterized by the "4Vs" of Big Data: Volume, Velocity, Variety, and Veracity. These dimensions increase the complexity of handling data, requiring innovative solutions for storage, processing, and analysis [3].

Despite advancements in database technologies, balancing technical performance and economic feasibility remains a significant challenge [4]. Traditional database

[*]Corresponding author.

E-mail address: turgay.bilgin@btu.edu.tr (T. T. Bilgin)

Received 09 October 2024; Received in revised form 23January 2025; Accepted 10 February 2025

management systems (DBMSs) often fall short in addressing the demands of modern data environments, particularly in scenarios where real-time data processing is essential [5]. Microsoft SQL Server (MSSQL) has emerged as a reliable platform for managing complex data operations due to its advanced querying capabilities, indexing methods, and robust data processing mechanisms [6]. However, its adaptability and cost-effectiveness in Big Data applications continue to be key areas of investigation.

One underexplored aspect of Big Data management is the role of data normalization in enhancing database performance. Data normalization minimizes redundancy and improves data integrity, making it a foundational principle in database design [7]. However, its application in high-velocity data environments, such as those requiring real-time processing, has not been sufficiently addressed. Current research often prioritizes technical innovations like query optimization, indexing strategies, and hardware upgrades [8]. Yet, the potential of normalization to improve performance while conserving resources remains inadequately examined.

Another critical gap involves the economic implications of database-dependent software systems. Organizations adopting such systems often face challenges related to vendor lock-in, limited database portability, and higher long-term maintenance costs [9]. These constraints can hinder flexibility and scalability, especially for businesses transitioning between database platforms. Addressing this issue requires database-independent software development strategies that ensure seamless integration across different systems while maintaining cost-effectiveness and operational flexibility [10].

This study aims to address these gaps by exploring how MSSQL can be optimized using data normalization techniques and how database-independent software design can alleviate economic burdens. The research hypothesizes that combining MSSQL's technical strengths with effective normalization strategies can significantly enhance performance in Big Data environments. Furthermore, adopting database-independent approaches can improve cost-efficiency and scalability by enabling interoperability across various DBMSs.

To test these hypotheses, this study examines data collected in milliseconds from 45 TAGs in a cable manufacturing plant. This high-velocity data environment represents the challenges of managing and analyzing large-scale, real-time production data. By implementing data normalization techniques within MSSQL, the research evaluates the impact on performance, scalability, and resource utilization. Additionally, it investigates the advantages of database-independent software in reducing costs and ensuring adaptability in industrial settings.

The findings of this research aim to provide actionable insights into balancing technical and economic considerations in Big Data management. By focusing on normalization as a cost-effective optimization tool and database independence as a strategy for flexibility, the study contributes to both academic literature and practical applications. These insights are particularly relevant for industrial environments, where real-time data analysis is essential for maintaining production quality and operational efficiency.

This study not only addresses gaps in the current literature but also offers practical solutions for organizations grappling with the complexities of Big Data. By combining theoretical exploration with real-world application, it provides a comprehensive perspective on effective strategies for managing large-scale data operations.

2 Literature Review

Several effective techniques have been identified for optimizing SQL Server performance in Big Data environments [5]. Key strategies include indexing, partitioning, sharding, and caching, which collectively enhance query performance and resource utilization [15]. Advanced query optimization methods, such as multi-level indexing and query rewriting, have achieved significant success by reducing data access and execution times by approximately 40% and 35%, respectively [6]. Additionally, implementing a query caching mechanism can further improve data access performance by prioritizing frequently used data and reducing execution times [17]. Innovative approaches, such as genetic algorithms, have demonstrated adaptability and efficiency in managing complex queries in the context of Big Data, effectively optimizing query performance [18]. While these techniques offer significant improvements, their implementation may vary depending on specific database architectures and workloads, necessitating a tailored approach to achieve optimal results [20]. In conclusion, a versatile strategy that combines these techniques is crucial for effectively managing the challenges SQL Server faces in Big Data environments.

Database performance optimization, particularly in SQL-based systems, has been extensively studied. Myalapalli et al. (2015) examined various SQL tuning techniques, such as indexing and query optimization, demonstrating their significant impact on reducing query execution times in large-scale databases [15]. Pedrozo and Vaz (2014) developed a tool for automatic index selection, which is critical in database management systems handling large volumes of data [16]. This tool optimizes query performance by automatically selecting the most efficient indexes based on query patterns, thereby significantly reducing computational load.

In the field of data warehousing, Correia et al. (2018) discussed the implementation of Fast Online Analytical Processing (OLAP) techniques in Big Data environments [17]. Their study emphasizes the necessity of real-time data processing capabilities for timely decision-making processes in large enterprises. They also address the challenges of maintaining high performance in the context of large data volumes. Sulistiani et al. (2020) explored the application of Agile methodologies in the development of OLAP systems for sales data analysis [18]. Their work demonstrated the effectiveness of the Agile approach in adapting to changing business needs and improving overall system response times.

The debate between normalization and denormalization in database design remains a critical topic. Erdinc et al. (2018) analyzed the impact of database design on software performance by comparing normalization and denormalization strategies [19]. They found that normalization reduces data redundancy and ensures data integrity; however, denormalization can significantly enhance query performance in certain scenarios by reducing the need for complex join operations [21]. Milicev (2021) introduced the concept of hyper models for the denormalization of transactional relational databases, offering a flexible framework for balancing performance and data consistency [22]. This approach provides a versatile solution for optimizing databases in environments where performance is a critical factor.

In addition to these studies, Alshemaimri et al. (2021) conducted a comprehensive survey on problematic database code snippets in software systems, highlighting how these issues negatively impact database performance [20]. Their work underscores the importance of code quality and its direct effect on database efficiency.

The role of SQL in big data processing has extended beyond the limitations of traditional databases to meet modern data analytics requirements. Silva et al. (2016) analyzed the transition of SQL from traditional databases to big data platforms and examined the challenges that arise during this process. The importance of techniques developed to optimize SQL for large-scale data workloads was emphasized [28].

A study by Santos et al. (2017) focused on the performance of SQL-on-Hadoop solutions. It examined how data warehouses could be optimized on low-performance hardware using SQL-on-Hadoop. The study demonstrated that effective data processing is achievable even on cost-effective infrastructures through SQL query optimization [26].

Real-time data analytics has emerged as a critical requirement in big data environments. Yang et al. (2014) introduced Druid, a real-time analytical data store designed to enhance speed and efficiency in query processing. The study provided significant insights into managing and analyzing high-velocity data streams [27].

Rahman et al. (2024) explored advanced optimization techniques to improve real-time query performance in SQL databases for big data analytics. The study evaluated the impact of indexing, query caching, and parallel processing techniques on handling large-scale data workloads [29].

Optimizing SQL in big data processing plays a crucial role in enhancing system performance. Uzzaman et al. (2024) reviewed best practices and techniques for optimizing SQL databases to handle big data workloads. The study highlighted innovative approaches in storage management and query planning [30].

Panwar (2024) proposed the use of advanced stored procedures to improve the capacity of SQL Server for processing big data. The study demonstrated how stored procedures optimize query operations, reducing both processing time and resource consumption under heavy data loads [32].

Performance comparisons of big data analytics platforms have been widely discussed in the literature. Pirzadeh et al. (2017) conducted a comprehensive evaluation of various big data analytics platforms. The study highlighted the advantages and limitations of SQL-based systems in query processing [31].

Ordonez (2013) questioned whether big data analytics could be performed directly within a DBMS and discussed the advantages of such an approach. The study noted that performing analytics at the database level reduces data transfer costs but poses challenges in performance optimization [33].

3 Methodology

This study utilized a comprehensive approach to examine Big Data management strategies within a cable manufacturing environment, focusing on the capabilities of Microsoft SQL Server. Data collection was carried out using OPC technology, which captured information from 45 TAGs across various production stages at a millisecond-level frequency [3][8]. The system architecture ensured efficient and reliable data transfer from production sensors to the MSSQL database, enabling seamless data ingestion and storage [5]. The dataset was carefully selected based on its relevance to critical production parameters, such as temperature, speed, and diameter. These factors were chosen for their importance in monitoring production quality and operational efficiency. The selection process also aimed to ensure the dataset’s representativeness, making it applicable to other industrial scenarios.

The sensors are integrated with the machine and transmit data to the database via the integrated KEPSERVER using OPC technology. A Microsoft service within the database continuously monitors the KEPSERVER and transfers millisecond-level records to the database. As shown in Figure 1, signals obtained from the machine are transferred to the KEPSERVER. A Windows service, specifically designed to transfer OPC data to the database located on the local server, continuously monitors the KEPSERVER and transfers any changed tag values to the database. This structure is as seen in figure 1.

A structured data normalization process was implemented to enhance database performance and maintain data integrity. Two techniques were primarily employed: z-score normalization and min-max scaling. Z-score normalization was used to standardize variables by centering them around the mean and scaling to unit variance, which is particularly useful for data following a Gaussian distribution. Min-max scaling, on the other hand, rescaled data to a range of [0, 1], making it suitable for bounded datasets. These methods ensured uniformity across the dataset while preserving relationships between variables. Additionally, denormalization was selectively applied to improve query efficiency in read-intensive scenarios, such as generating reports. This approach balanced the benefits of normalization, including reduced redundancy and improved consistency, with the performance advantages of denormalized structures for specific use cases.

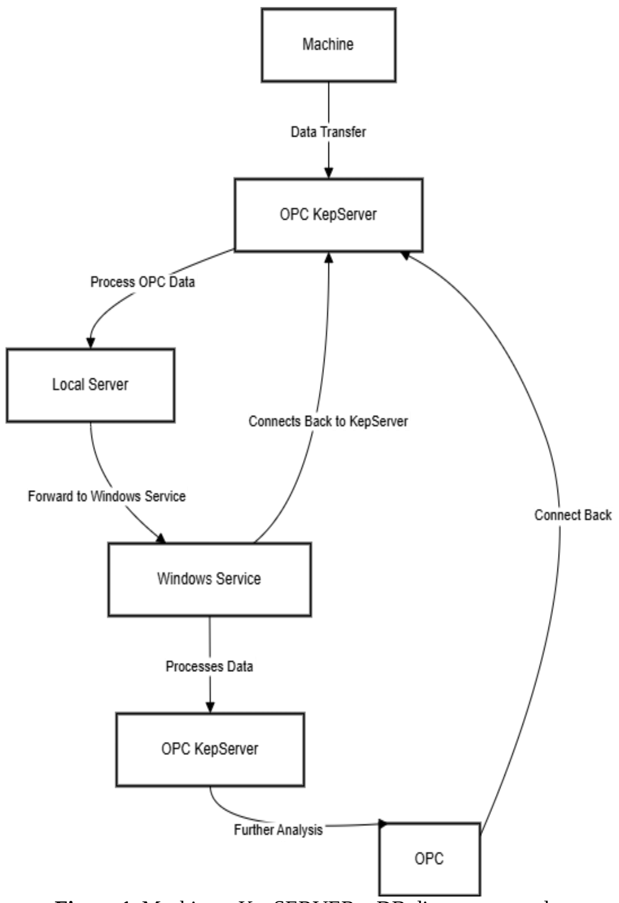


Figure 1. Machine – KepSERVER – DB diagram example.

The data processing phase involved the design and execution of optimized SQL queries to group and analyze the collected data [13]. These queries were tailored to extract statistical metrics, such as mean values, variances, and trend patterns, which are essential for monitoring production processes. Performance improvement techniques, including the use of indexed views, data partitioning, and execution plan analysis, were applied to minimize query execution times and optimize system resource usage.

To address challenges related to vendor lock-in and database portability, a database-independent software development strategy was adopted. This approach involved creating abstraction layers to decouple application logic from database-specific functionalities, allowing compatibility with various DBMS platforms. By implementing this strategy, the study demonstrated enhanced flexibility and potential cost savings, which are critical for scalable Big Data solutions.

The evaluation of the implemented methods relied on performance metrics such as query execution times, CPU usage, and memory consumption. These metrics provided measurable insights into system efficiency. Security protocols, including access controls and data encryption, were incorporated to ensure data reliability. Moreover, data integrity checks, such as consistency constraints and validation rules, were performed to maintain accuracy. Statistical analyses were conducted on the processed data to extract meaningful insights. Techniques such as regression analysis and anomaly detection were utilized to identify trends and outliers, contributing to improved production quality and decision-making.

To ensure transparency and reproducibility, future research should provide more detailed technical explanations of the normalization processes and algorithms employed. For instance, specifying the formulas, software tools, or custom scripts used would enhance methodological clarity. Additionally, further elaboration on the criteria for dataset selection could improve the study’s generalizability to other industrial contexts. By addressing these aspects, this research contributes to advancing practical Big Data management strategies while balancing technical efficiency with economic feasibility.

3.1 Data Collection Process

The data collection process, a critical component of this study, was designed to ensure accurate, reliable, and continuous data gathering from various stages of the cable production line [8]. This process utilizes OPC (OLE for Process Control) technology, a standard that facilitates data exchange between devices and software in industrial automation systems [9].

OPC technology is used to collect data from numerous sensors, measurement devices, and control systems on the production line [10]. This approach enables the consolidation of data from different production lines and devices into a single centralized Microsoft SQL Server (MSSQL) database [11]. The OPC server regularly pulls data from the sensors and devices, and a Windows service on the OPC server ensures that this information is recorded in the MSSQL database in a specified format [12].

Data is collected from 45 TAGs in milliseconds, capturing critical parameters of the cable production process [8]. These parameters are essential for the efficient monitoring and analysis of the production process [15].

These data points are collected from various stages of the production process, reflecting the current state of the system. The collected data is critical for monitoring and optimizing the production line, identifying potential issues, and conducting quality control procedures.

Data collection intervals, data volume, and collection methods are key factors that influence the overall performance of the system [7]. These factors are carefully evaluated to balance the need for comprehensive data with system efficiency and resource utilization [8]. This data collection process forms the foundation for subsequent data processing, analysis, and performance optimization steps in the study, enabling a comprehensive investigation of Big Data management in the context of cable production [11].

Table 1. Key data collected during cable production process by chamber.

Data	Description	Chamber
Metre	Measures the length of the produced cable and is used to monitor the total production quantity. This data is critical for determining how long the production line has been operational and the volume of products manufactured.	1
HatHizi (Line Speed)	Monitors the speed of the cable production line. This data is used to evaluate the efficiency of the production process and ensure that the line operates at an optimal speed.	1
SicakCapReel (Hot Diameter)	Measures the diameter of the cable during production. This data is an important factor influencing the cable's physical properties and final quality.	1
SogukCapReel (Cold Diameter)	Monitors the diameter of the cable after production. This measurement ensures that the final dimensions and shape stability of the cable are maintained.	1
Spark	Tracks sparks that occur during production. This data helps detect electrical issues in the production process.	1
VidaDevri (Screw RPM)	Measures the rotational speed of the screw mechanism used in the cable production line. This data directly affects the cable's quality and is monitored to ensure process stability.	1
1BolgeReel - 6BolgeReel	Monitors the measurements in different regions of the cable production line. These data are essential for tracking and optimizing changes occurring in various regions of the production line.	2

3.1.1 Implementation of OPC Technology

In this study, the implementation of OPC (OLE for Process Control) technology constitutes a critical component of the data collection process [9]. OPC serves as a standardized interface for industrial automation systems, facilitating seamless communication between various hardware devices and software applications involved in the cable production process [12].

In our setup, OPC technology is used to establish a robust and efficient data collection system [10]. The implementation consists of several key components:

- **OPC Server:** A dedicated OPC server was set up to mediate between the physical devices on the production line and the data collection software. This server is responsible for reading data in real-time from various sensors, measurement devices, and control systems [13].
- **Device Integration:** Each of the 45 TAGs designated for data collection is configured to communicate with the OPC server [8]. These TAGs represent different parameters at various stages of production, such as cable length, line speed, hot diameter, and cold diameter [14].
- **Data Formatting:** The OPC server is programmed to format the collected data into a standard structure compatible with the Microsoft SQL Server database. This ensures consistency in data representation and facilitates ease of processing and analysis in subsequent stages [11] [16]. A Windows service on the OPC server accesses the SQL Server and is programmed to record the data in a structure that fits the database architecture. This minimizes resource usage on the OPC server [12].
- **Communication Protocol:** The implementation uses standard OPC protocols such as OPC DA (Data Access) or OPC UA (Unified Architecture), depending on the specific requirements of the devices and systems in use [9] [10]. These protocols ensure reliable and secure data transmission from the production site to the database [11].
- **Data Buffering:** To address potential network interruptions or database outages, a data buffering mechanism is included in the OPC implementation [12]. This prevents data loss during temporary communication failures [15].
- **Time Synchronization:** The OPC server is synchronized with a central time source to provide accurate timestamps for all collected data points. This is crucial for maintaining data integrity and enabling time-based analyses [8].
- **Security Measures:** The OPC implementation includes security features such as encryption and authentication to protect the data transmission process from unauthorized access or interference [17].
- **Scalability:** The OPC configuration is designed to be scalable, allowing new TAGs or devices to be added easily as the production system expands or evolves [18].

By leveraging OPC technology in this manner, we establish a reliable and efficient data collection infrastructure that forms the foundation of our Big Data management strategy [10]. This implementation enables us to capture comprehensive and real-time data from the cable production process, providing the raw data necessary for subsequent processing and analysis stages [8] [14].

3.1.2 Data Collection in Cable Manufacturing

The data collection process in cable manufacturing is designed to capture comprehensive information on various aspects of the production process [7]. This process aims to provide a holistic view of cable manufacturing operations by gathering data from critical points along the production line. The data collection system utilizes 45 TAGs distributed across these production areas. Each TAG corresponds to a specific sensor or measurement device on the production line. These TAGs are configured to transmit data at regular intervals, ensuring a continuous flow of information about the production process [12].

Data flows into the system in real-time in millisecond format, contributing to the formation of Big Data [13]. The collected data is instantly transmitted to the centralized Microsoft SQL Server database via the OPC infrastructure. This real-time data transfer allows production managers and analysts to access up-to-date information about the production process.

Various measures are implemented to ensure data integrity and reliability:

- **Sensor Calibration**: Regular calibration of sensors and measurement devices is essential to maintain accuracy [14].
- **Data Validation**: Automated checks are employed to identify and flag anomalous readings.
- **Redundancy**: Critical measurements are typically captured by multiple sensors to ensure data continuity in case of sensor failure.

This approach provides a rich dataset for subsequent analysis and optimization efforts. By collecting detailed information on production parameters, the system offers in-depth insights into production efficiency, product quality, and areas with potential for improvement within the cable manufacturing process.

3.1.3 Data Collection Intervals

In cable manufacturing, the effective management of Big Data systems critically relies on the determination and implementation of appropriate data collection intervals. This study adopts a carefully considered approach to data collection scheduling that balances the need for detailed information with system efficiency and resource management [11].

The primary data collection interval has been set to one minute. This interval was selected after considering several factors:

1. **Dynamics of the Production Process**: A one-minute interval is well-aligned with the typical rate of change in cable manufacturing parameters. It is frequent enough to capture significant fluctuations in production conditions while avoiding the omission of critical events.
2. **Data Volume Management**: Shorter intervals may provide more detailed data but also significantly increase the volume of collected data. A one-minute interval maintains a manageable level of data volume for storage and processing, while still providing adequate detail.
3. **System Resource Utilization**: More frequent data collection increases the load on both the OPC server and the SQL database. The chosen interval helps to optimize the use of system resources [15].
4. **Analysis Requirements**: A one-minute interval provides sufficient resolution for most analytical needs, including trend analysis, quality control, and production optimization.

However, the system is designed with the flexibility to adjust collection intervals based on specific production scenarios:

1. **Critical Process Stages:** During critical stages of production or when new processes are being tested, the system allows for a temporary increase in collection frequency, potentially reducing intervals to as short as one minute.
2. **Low Activity Periods:** During periods of low production activity or machine downtime, the collection interval can be extended to reduce unnecessary data accumulation [18].
3. **Event-Triggered Collection:** In addition to regular intervals, the system is configured to capture data instantaneously when specific thresholds are exceeded or particular events occur. This ensures that critical information is not missed between regular collection points.
4. **Adaptive Intervals:** The system includes an adaptive mechanism that automatically adjusts collection intervals based on the rate of change in key parameters. This allows for more frequent data collection during periods of high variability and less frequent collection during stable periods [17].

The implementation of these data collection intervals includes:

1. **Configuration of the OPC Server:** Configuring the OPC server to retrieve data from sensors at specified intervals.
2. **Synchronization of Data Communication:** Ensuring that data transmission to the SQL database is synchronized with these intervals.
3. **Timestamping:** Applying timestamps to each data point to ensure accurate temporal analysis.

By carefully managing the data collection intervals, the system ensures comprehensive and efficient capture of production data. This approach facilitates detailed and reliable monitoring and analysis of the cable production process.

3.1.4 System Architecture

The architecture of the data collection system is designed to ensure efficient collection, transmission, and storage of data from the cable production process. This system integrates various components to provide a seamless flow of data from the production line to analytical tools. The architecture consists of the following key components:

1. **Sensors and Measurement Devices:** These devices are distributed along the production line and capture various parameters such as cable length, line speed, hot diameter, and cold diameter. They serve as primary data sources that continuously monitor the production process.
2. **OPC Server:** This server acts as an intermediary between physical devices and the database. It collects data from sensors and measurement devices and standardizes the data format to ensure consistent data representation.
3. **Windows Service:** This service is located on the OPC server and is designed to write the standardized data to the SQL Server. It continuously performs connection checks and validations. In the event of a disruption, it temporarily stores data in a local directory on the OPC server at one-minute intervals to maintain data integrity. Once the connection is reestablished, it sequentially reads the data and continues writing to the SQL Server.
4. **Microsoft SQL Server Database:** This is the central repository where all collected data is stored. It is designed to handle large volumes of data efficiently, supporting rapid data retrieval and complex queries.
5. **Monitoring and Control Interface:** This component provides a user-friendly interface for real-time monitoring of the production process. It allows operators and managers to view current production parameters and historical data.

In this architecture, the data flow follows a specific path:

1. **Data Origin:** Data originates from sensors and measurement devices on the production line.
2. **Data Collection:** The OPC server collects this data at regular intervals (typically every minute).
3. **Data Transmission:** The collected data is then transmitted to the Microsoft SQL Server database for storage.
4. **Data Retrieval:** The monitoring and control interface retrieves data from the database to display real-time and historical information.

This architecture is designed with several key features:

1. **Scalability:** The system can accommodate additional sensors or increased data collection rates as production needs evolve [18].
2. **Reliability:** Redundancy measures are in place to ensure continuous data collection even in the event of component failures.
3. **Security:** Data transmission and storage incorporate encryption and access control measures to protect sensitive production information [17].
4. **Performance Optimization:** The architecture is optimized to handle high-frequency data collection and storage operations without compromising system performance.
5. **Integration Capability:** The system is designed to easily integrate with other enterprise systems, such as ERP or quality management software, to facilitate comprehensive data analysis and decision-making processes [20].

This robust system architecture provides efficient and reliable data collection in the cable manufacturing environment, forming the foundation for effective Big Data management. By enabling real-time monitoring, historical analysis, and data-driven decision-making processes, it contributes to improvements in production efficiency and quality control.

3.2 Data Processing

The data processing phase is one of the fundamental components of this study and focuses on transforming raw data into meaningful and actionable information [5]. This process leverages the capabilities of Microsoft SQL Server to efficiently handle large volumes of data and derive valuable insights from the cable manufacturing process [13].

At the core of data processing is a complex SQL query structure that performs multiple operations simultaneously. This query is designed to process data collected from various stages of the production line, such as Chamber 1 and Chamber 2, and to generate comprehensive production metrics for each reel [6]. The key components of data processing are given below:

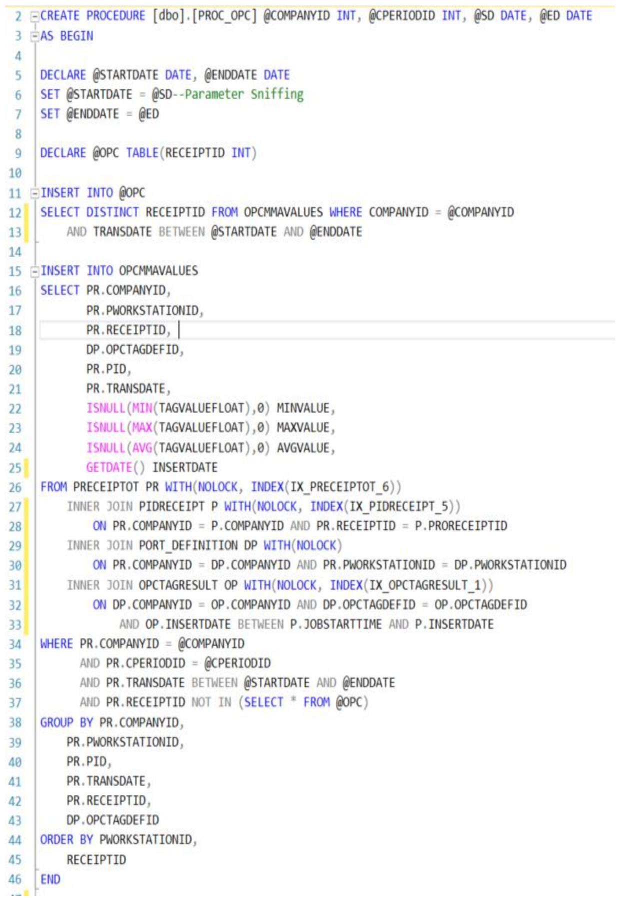
1. **Data Relational Mapping:** The query establishes relationships between different tables in the database, particularly focusing on the relationships between the "Reel" and "ReelReadings" tables [12]. This relationship is based on matching the ReelID values and ensures the aggregation of all relevant data for each reel.
2. **Data Grouping:** Data is grouped by ReelID, allowing for the analysis of production metrics for each reel. This grouping is essential for understanding the performance and characteristics of each production unit.
3. **Statistical Calculations:** For each reel, the query computes several statistical measures:
   - Minimum, maximum, and average values of meter readings
   - Minimum, maximum, and average values of line speed
   - Minimum, maximum, and average values of hot diameter
   - Minimum, maximum, and average values of cold diameter
4. **Time-Based Filtering:** The query includes a time-based filter, typically using the StartDate and

EndDate fields from the Reel table, to focus on specific production periods [16].

5. **Data Aggregation:** Results are consolidated to provide a comprehensive view of production metrics for each reel.

The SQL query utilized in the data processing phase is designed to group the data collected during cable production meaningfully and perform various statistical calculations. This query establishes relationships between different tables, integrates relevant data, and extracts key statistical information such as minimum, maximum, and average values. The primary aim of the query is to provide detailed insights into the various stages of the production process and the events occurring at these stages by conducting statistical analyses on the collected production data.

This data processing procedure facilitates in-depth analyses of the different stages of the cable production process and enables an understanding of the performance of these stages. The statistical information obtained is crucial for optimizing the production process and for quality control.



```sql
CREATE PROCEDURE [dbo].[PROC_OPC] @COMPANYID INT, @CPERIODID INT, @SD DATE, @ED DATE
AS BEGIN

DECLARE @STARTDATE DATE, @ENDDATE DATE
SET @STARTDATE = @SD--Parameter Sniffing
SET @ENDDATE = @ED

DECLARE @OPC TABLE(RECEIPTID INT)

INSERT INTO @OPC
SELECT DISTINCT RECEIPTID FROM OPCMMAVALUES WHERE COMPANYID = @COMPANYID
    AND TRANSDATE BETWEEN @STARTDATE AND @ENDDATE

INSERT INTO OPCMMAVALUES
SELECT PR.COMPANYID,
       PR.PWORKSTATIONID,
       PR.RECEIPTID,
       DP.OPCTAGDEFID,
       PR.PID,
       PR.TRANSDATE,
       ISNULL(MIN(TAGVALUEFLOAT),0) MINVALUE,
       ISNULL(MAX(TAGVALUEFLOAT),0) MAXVALUE,
       ISNULL(AVG(TAGVALUEFLOAT),0) AVGVALUE,
       GETDATE() INSERTDATE
FROM PRECEIPTOT PR WITH(NOLOCK, INDEX(IX_PRECEIPTOT_6))
    INNER JOIN PIDRECEIPT P WITH(NOLOCK, INDEX(IX_PIDRECEIPT_5))
        ON PR.COMPANYID = P.COMPANYID AND PR.RECEIPTID = P.PRORECEIPTID
    INNER JOIN PORT_DEFINITION DP WITH(NOLOCK)
        ON PR.COMPANYID = DP.COMPANYID AND PR.PWORKSTATIONID = DP.PWORKSTATIONID
    INNER JOIN OPCTAGRESULT OP WITH(NOLOCK, INDEX(IX_OPCTAGRESULT_1))
        ON DP.COMPANYID = OP.COMPANYID AND DP.OPCTAGDEFID = OP.OPCTAGDEFID
           AND OP.INSERTDATE BETWEEN P.JOBSTARTTIME AND P.INSERTDATE
WHERE PR.COMPANYID = @COMPANYID
     AND PR.CPERIODID = @CPERIODID
     AND PR.TRANSDATE BETWEEN @STARTDATE AND @ENDDATE
     AND PR.RECEIPTID NOT IN (SELECT * FROM @OPC)
GROUP BY PR.COMPANYID,
    PR.PWORKSTATIONID,
    PR.PID,
    PR.TRANSDATE,
    PR.RECEIPTID,
    DP.OPCTAGDEFID
ORDER BY PWORKSTATIONID,
    RECEIPTID
END
```

Figure 2. Analysis query.

To understand the operation of the SQL query, it is important to examine the functions of the tables used and the logic of the query. Below is a detailed explanation of these tables and the query logic:

- **PRECEIPTOT (PR) Table**: This table contains transaction records related to the production process. Fields such as PR.COMPANYID, PR.PWORKSTATIONID, PR.RECEIPTID, and PR.TRANSDATE indicate the company, workstation, and time period in which the production process occurred. This table plays a central role in the data processing phase.
- **PRECEIPTOM (PM) Table**: This table is linked to the PR table and contains details of the processes within the production workflow. The PR.COMPANYID and PR.RECEIPTID fields establish the relationship between these two tables.
- **STOCK (S) Table**: This table provides information about the inventory used in production. It is connected to the PM table and provides details about the materials used during the production process.
- **PORT_DEFINITION (DP) Table**: This table contains definitions of the data collected via OPC. The PR.COMPANYID and PR.PWORKSTATIONID fields link this table to the PR table. The DP.OPCTAGDEFID field specifies a unique identifier for each measurement point.
- **OPCTAGRESULT (OP) Table**: This table includes real-time data collected through OPC. The DP.OPCTAGDEFID and OP.INSERTDATE fields link this table to the DP table. It stores data collected within specific time periods.

The query combines records from the PR table with corresponding records in the PM, S, DP, and OP tables. This join operation is performed using the INNER JOIN command. INNER JOIN ensures that only records with matches in both tables are retrieved, allowing for the processing of relevant data only.

One of the primary functions of the query is to calculate the minimum, maximum, and average values of data collected within a specific time frame (between PR.WORKSTARTTIME and PR.INSERTDATE). These statistical calculations make the data more meaningful and enable the evaluation of performance at various stages of the production process.

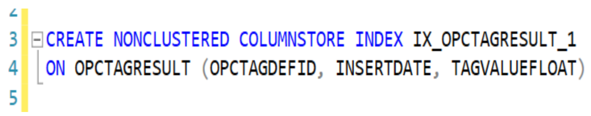
An important phase of the query is data grouping. This grouping process allows for the aggregation of data based on specific fields and facilitates statistical analysis within these groups. The GROUP BY clause used in the query enables the grouping of data by fields such as PR.COMPANYID, PR.PWORKSTATIONID, PR.PID, PR.TRANSDATE, PR.RECEIPTID, and DP.OPCTAGDEFID. This grouping allows for the segregation of data from different stages of the production process and makes it possible to perform comparisons between these stages. For example, comparing data collected at different time intervals on the same production line can help identify deviations in the production process and determine the causes of these deviations.

The processed data provides valuable insights into production efficiency, quality control, and areas with potential for improvement in the cable manufacturing process. Additionally, it enables the identification of trends, anomalies, and correlations among various production parameters. These insights contribute to optimizing the production process, addressing quality issues, and enhancing overall operational efficiency.

3.3 Query Optimization Techniques

Optimizing the data processing phase is crucial for efficiently handling large data sets and maximizing system performance. To minimize lock contention and enhance concurrency, the WITH(NOLOCK) hint has been strategically employed in queries. This technique allows read operations to bypass locks held by other transactions, significantly reducing query wait times. By decreasing database locking, this method facilitates faster query execution and improves overall system performance. However, it is important to note that this approach might occasionally read uncommitted data. This trade-off is considered acceptable in real-time monitoring scenarios where immediate data access is more critical than absolute consistency.

A comprehensive indexing strategy has been implemented to optimize query performance. Key indexes such as IX_PRECEIPTOT_6 and IX_OPCTAGRESULT have been created on frequently queried columns as shown in Figure 3. These indexes are regularly maintained and adjusted according to query patterns and performance metrics.



```
CREATE NONCLUSTERED COLUMNSTORE INDEX IX_OPCTAGRESULT_1
ON OPCTAGRESULT (OPCTAGDEFID, INSERTDATE, TAGVALUEFLOAT)
```

Figure 3. Create index.

For fact tables with a large number of rows, columnstore indexes have been applied. These indexes provide significant performance improvements for analytical queries that scan large sections of the table.

4. Performance Analysis

Efficient data processing in industrial environments managing large data sets is critical due to its direct impact on operational efficiency and cost-effectiveness. In the context of cable production, reducing data processing times enhances both production line efficiency and overall system performance. This section highlights significant performance improvements achieved through normalization and other optimization techniques by examining their impact on data processing.

4.1 Impact of Normalization on Data Processing

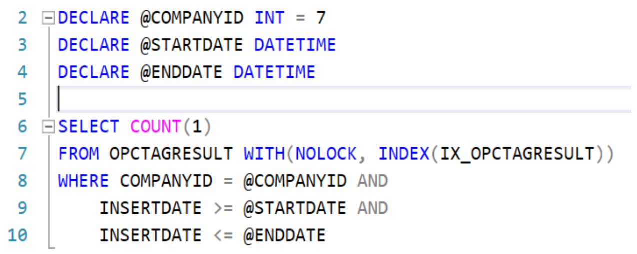
At the beginning of the data processing phase, the system collected data from 45 TAGs at one-minute intervals, resulting in approximately 13,500 records per interval. During peak production periods, data was collected from 81 machines and 824 TAGs, creating a substantial data load. On a single day, this data extraction process accumulated approximately 56 million rows in the database, placing significant pressure on system resources.

According to Table 2, as the number of records increased from 10,000 to 56 million, the corresponding size of the dataset grew from 551 MB to over 3 TB, further intensifying the pressure on the system.

Table 2. Growth of Dataset Size and Record Count Impact on System Load

Sequence ID	Record Count	Size (megabyte)
1	10.000	551
2	25.000	1380
3	50.000	2765
4	100.000	5546
5	200.000	11136
6	300.000	16815
7	400.000	22622
8	500.000	28595
9	1.000.000	57647
10	1.500.000	87508
11	2.000.000	118428
12	2.500.000	150640
13	3.000.000	184383
14	4.000.000	250638
15	5.000.000	320316
16	6.000.000	393989
17	7.000.000	944520
18	12.000.000	1111568
19	25.000.000	1290530
20	56.000.000	3000000

The total data processing time extended to approximately 40 minutes and 45 seconds, illustrating the challenges of processing such large datasets in real-time (refer to Figure 4 for the SQL query used in the pre-normalization data analysis).

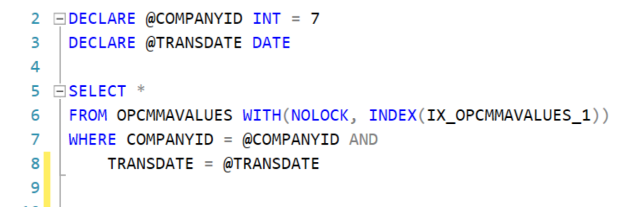


```
DECLARE @COMPANYID INT = 7
DECLARE @STARTDATE DATETIME
DECLARE @ENDDATE DATETIME

SELECT COUNT(1)
FROM OPCTAGRESULT WITH(NOLOCK, INDEX(IX_OPCTAGRESULT))
WHERE COMPANYID = @COMPANYID AND
    INSERTDATE >= @STARTDATE AND
    INSERTDATE <= @ENDDATE
```

Figure 4. Pre-normalization query.

To address these inefficiencies, a comprehensive normalization process was implemented. Normalization significantly reduced the dataset size by eliminating redundancy and unnecessary data points. For every 1-minute interval, the number of rows decreased from 13,500 to just 45. This reduction dramatically improved query performance, reducing the daily dataset to only 283 rows and lowering the query processing time to an impressive 0.1 seconds (see Figure 5 for the SQL query used in the post-normalization analysis). The application of normalization clearly demonstrated its ability to optimize large-scale data processing, resulting in faster query times and more efficient system resource usage.



```
DECLARE @COMPANYID INT = 7
DECLARE @TRANSDATE DATE

SELECT *
FROM OPCMMAVALUES WITH(NOLOCK, INDEX(IX_OPCMMAVALUES_1))
WHERE COMPANYID = @COMPANYID AND
    TRANSDATE = @TRANSDATE
```

Figure 5. Post-Normalization query.

4.2 Performance Improvements

Normalization led to significant performance enhancements overall [19]. Before normalization, processing the dataset consisting of 56 million rows (as indicated in Table 1, over 3 TB) took more than 40 minutes, causing substantial delays in critical operations. After normalization, the reduced dataset allowed query processing times to fall below one second. This dramatic reduction in both data size and processing time underscores the value of normalization as a performance optimization strategy [21].

In addition to faster query times, the reduction in data size also optimized disk I/O operations, which are crucial in large-scale databases [22]. Normalization enabled more efficient data retrieval, thereby improving overall system performance and reducing storage demands. As a result, the system was able to achieve higher query throughput without performance degradation, thereby enhancing scalability and operational capacity.

4.2.1 Resource Utilization and System Optimization

Normalization and optimization techniques have significantly improved query performance while enhancing resource utilization. By eliminating redundant and repetitive data, the load on critical resources such as CPU, memory, and disk I/O has been substantially reduced [20]. These improvements have made data processing more efficient and minimized the operational burden on the system. In industrial environments requiring high data processing capacities, such as cable manufacturing, these advancements have directly contributed to maintaining production efficiency and reducing operational costs.

As shown in Figure 6, prior to normalization, the database often experienced performance bottlenecks caused by excessive resource consumption due to large datasets. For example, during query execution, CPU usage averaged 88%, memory usage reached 84%, and disk I/O utilization was negligible at 1 MB. However, as demonstrated in Figure 7, significant improvements were observed after normalization: CPU usage decreased to 20%, memory usage dropped to 37%, and disk I/O utilization slightly increased to 6 MB. These measurements were obtained using SQL Server Profiler, clearly indicating improved resource utilization.

These optimizations have enabled the system to process data more effectively, reduced the risk of resource exhaustion, and increased its capacity to handle growing data volumes. Furthermore, the system's scalability has been enhanced without requiring additional resources, contributing to its long-term sustainability.

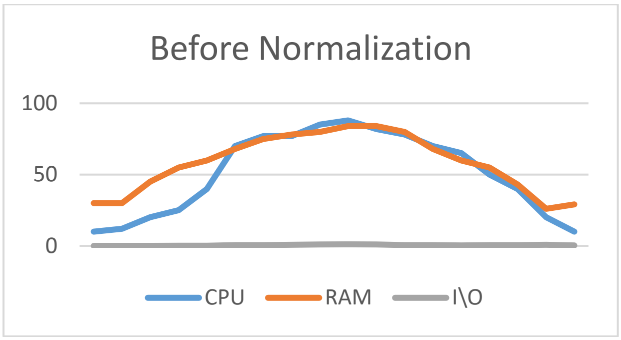


Figure 6. Before Normalization – Resource Usage

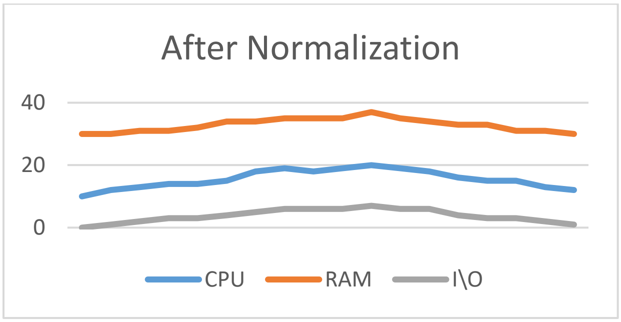


Figure 7. Before Normalization – Resource Usage

4.2.2 Reduction in Query Times

Normalization has had a profound impact on reducing query execution times. Initially, querying large datasets could take up to 40 minutes, significantly affecting the system's response capability and real-time data acquisition. After the implementation of normalization, the reduction in data size directly led to a significant decrease in query times.

Table 3 presents a comparison of query execution times before and after normalization. For instance, prior to optimization, querying 1 million records took approximately 12 seconds, whereas after optimization, the same query was processed in just 0.91 seconds. This dramatic reduction in query time demonstrates the effectiveness of normalization and other optimization techniques in enhancing system performance.

Table 3. Execution times of SQL queries.

Sequence ID	Before Norm. (secs)	After Norm. (secs)
1	0,4	0,001
2	0,5	0,003
3	1,3	0,005
4	1,4	0,008
5	2,6	0,011
6	3,3	0,016
7	4,1	0,032
8	5,2	0,04
9	8,4	0,82
10	12	0,91
11	14,6	1,02
12	16,7	1,13
13	22,3	1,54
14	29,5	1,82
15	39,9	2,01
16	44,5	2,26
17	51,1	2,41
18	289	2,92
19	612	3,47
20	1835	3,88

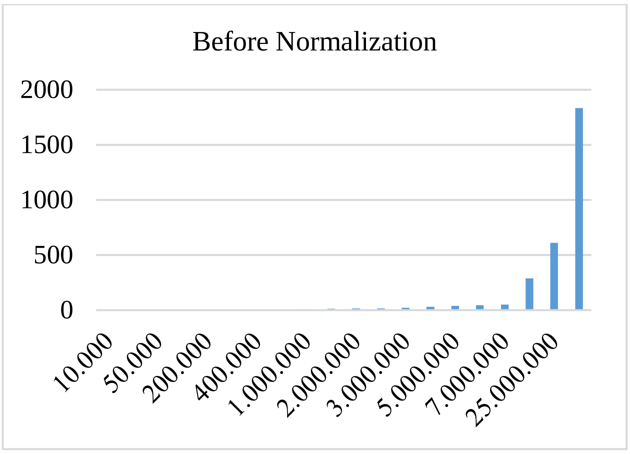


Figure 8. Before normalization.

Figure 8 further illustrates the stark difference in query performance before and after optimization. The system's ability to handle more queries in parallel with reduced latency has increased overall data processing capacity and enabled the handling of larger datasets without compromising performance.

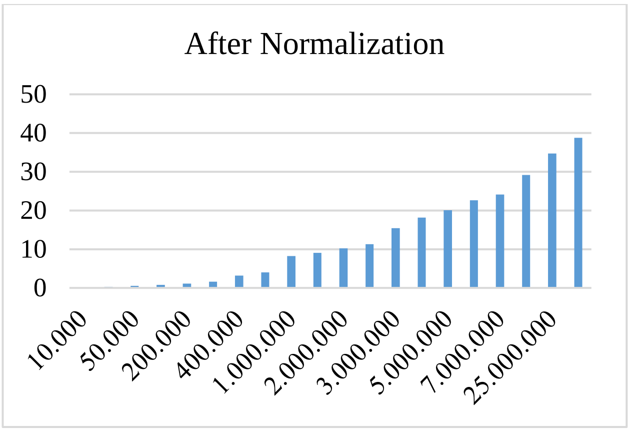


Figure 9. After normalization.

The reduction in query times is crucial in real-time production environments, as timely data analysis is essential for decision-making and process optimization [17]. Faster query execution also enables the system to handle multiple queries simultaneously, thereby enhancing overall operational efficiency.

4.3 Strategic Importance of Normalization

In this study, normalization has enabled the system to allocate resources more efficiently by reducing the overall dataset size, which has directly contributed to faster processing times and improved resource utilization [19]. The long-term benefit of normalization lies in its ability to scale the system as data volumes increase, without compromising performance or requiring significant additional resources. As demonstrated in Table 2 and Figures 8 and 9, normalization has had a tangible impact on reducing query execution times and improving system operations.

However, normalization is not a one-time solution; it should be part of a continuous data management strategy [21]. As data sets evolve and expand, regular review and adjustment of the normalization process are necessary to ensure sustained performance gains. In dynamic industrial environments like cable production, this approach helps maintain operational efficiency, supports scalability, and lays the groundwork for more advanced data analysis techniques.

In conclusion, the strategic importance of normalization lies in its ability to optimize both short-term performance and long-term sustainability [22]. By integrating normalization as a fundamental part of the data management process, organizations can ensure that their systems remain responsive and scalable in the face of increasing data complexity.

5 Results and Recommendations

Effective data management is crucial for modern industrial processes, where the ability to process large datasets directly impacts operational efficiency and business competitiveness [23]. This study examined the optimization of database performance through normalization and indexing techniques on Microsoft SQL Server (MSSQL). The results highlight both the strengths and limitations of MSSQL in managing large-scale data and provide insights for its application in industrial environments.

One of MSSQL's key strengths is its advanced indexing capabilities, particularly the use of non-clustered indexes on fields such as DATE and TAGID. These indexes significantly reduce data access times, enhancing query performance. This feature is especially beneficial for managing frequent queries on large datasets. The flexibility of MSSQL in performance optimization, such as the WITH(NOLOCK) query hint, further improves efficiency by minimizing locking during query execution. This makes MSSQL a valuable tool for large-scale data processing.

MSSQL also excels in data security and integrity, offering robust tools for auditing, error management, and security. These features ensure consistent and reliable data storage, minimizing the risks of data loss or corruption. Its scalability allows MSSQL to grow with increasing data volumes, supporting long-term data management strategies in expanding industrial environments. This scalability is particularly relevant for real-time data collection systems, such as those used in production lines, where large amounts of sensor data are generated continuously.

However, the study identifies several limitations of MSSQL. The high licensing costs of its advanced features pose a financial challenge, especially for small and medium-sized enterprises (SMEs). These costs can make it difficult for organizations with limited budgets to adopt MSSQL for large-scale applications. Additionally, as databases increase in size and complexity, MSSQL may experience performance degradation, requiring regular maintenance and optimization. Index fragmentation is a common issue that requires frequent monitoring. Fragmented indexes must be reorganized or rebuilt based on their fragmentation levels to maintain optimal query performance.

Another limitation is MSSQL's restricted support for distributed data management [24]. In scenarios requiring the management of large datasets across multiple geographic locations or systems, MSSQL’s limited capabilities in distributed databases and clustering solutions can hinder its effectiveness. Alternative platforms, such as cloud-based or open-source solutions, may offer more suitable options for such use cases. Exploring these alternatives could reveal opportunities for improving scalability and flexibility in distributed environments.

Normalization, while improving data integrity and reducing redundancy, presents its own challenges. It can introduce additional complexity and computational overhead during query execution, particularly in environments requiring real-time data access. These trade-offs, such as increased response times for complex queries or challenges in maintaining normalized forms during rapid data updates, should be acknowledged. A deeper understanding of these trade-offs could guide the development of hybrid approaches that balance normalization and denormalization to achieve optimal performance.

The findings of this study are not limited to the cable manufacturing sector but are applicable to other industries with similar production processes. In sectors such as healthcare, logistics, and finance, where real-time data collection and processing are critical, MSSQL’s optimization techniques can be adapted to meet specific operational needs. Key factors such as the volume of data collected, the number of tags (TAGs) used, and data collection intervals play a crucial role in determining the success of database management strategies. Further exploration of these industries could help generalize the findings and provide industry-specific recommendations.

With the increasing adoption of machine learning and advanced analytics, MSSQL's role in supporting these technologies is becoming increasingly important. Predictive modeling and analytics require clean, consistent, and enriched datasets. MSSQL’s compatibility with programming languages such as R and Python enables seamless integration of machine learning models directly into the database environment. Future research should explore how MSSQL can be optimized to streamline data preparation and enhance its support for advanced analytics workflows.

Dynamic optimization techniques that adapt to workload changes in real time could further enhance MSSQL’s performance. For example, machine learning algorithms could be employed to monitor query execution patterns and dynamically optimize indexes, queries, and data structures. These approaches could improve resource management and ensure consistent performance, even in high-demand environments.

Finally, MSSQL’s potential in federated learning scenarios should be explored. Federated learning allows for distributed data management while maintaining data privacy, making it a suitable option for industries where data security is critical. Additionally, MSSQL’s integration with visualization tools like Power BI and Tableau offers significant potential for enhancing reporting and decision-making processes.

This study highlights the strengths of MSSQL in indexing, query optimization, and data integrity while also addressing its limitations in cost, distributed management, and normalization trade-offs. Acknowledging the overhead and complexity introduced by normalization helps present a more balanced view. The insights gained provide a foundation for future research to explore alternative platforms, hybrid optimization techniques, and advanced analytics workflows. These advancements could lead to more scalable, adaptable, and cost-effective data management solutions for modern industrial applications.

5.1 General Evaluation of Our Processes

This study aimed to optimize MSSQL performance for managing large-scale data by employing data processing and normalization techniques. By simplifying data workflows, the approach successfully reduced processing times and enhanced overall system efficiency. The SQL queries designed for this purpose utilized non-clustered indexing, which significantly decreased data access times and improved query execution speeds, particularly in scenarios involving datasets with millions of rows. Among the optimization strategies applied, normalization emerged as the most impactful. By reducing redundancy, minimizing dataset size, and retaining only essential information, normalization not only enhanced data integrity but also optimized resource utilization. This led to shorter query times and more responsive system performance. However, the normalization process introduced certain trade-offs, such as increased complexity in real-time data access and added processing overhead, which need to be carefully considered when implementing such strategies.

5.2 Recommendations for Future Work

To sustain the benefits of normalization and indexing in MSSQL, continuous monitoring and optimization are essential. Index fragmentation is a common issue that can degrade query performance over time, particularly in large and dynamic databases. Regular maintenance is necessary to rebuild fragmented indexes and ensure consistent query efficiency. Similarly, normalization, which organizes data into structured formats, must be revisited periodically to adapt to evolving datasets and changing system requirements. This approach ensures that the performance improvements achieved during initial implementation are preserved and even enhanced.

Future studies should explore the applicability of these techniques to alternative database platforms. MSSQL, while effective in managing large datasets, has certain limitations in distributed data management and clustering environments. Research into open-source systems such as PostgreSQL or cloud-based solutions like Amazon Aurora and Google BigQuery could reveal how these platforms address similar optimization challenges. Comparative analyses would provide valuable insights into the scalability, flexibility, and cost-effectiveness of normalization and indexing across various systems. Additionally, studying hybrid approaches that balance normalization and denormalization could address some of the trade-offs identified in this study, particularly in scenarios requiring real-time data access.

As industries increasingly adopt machine learning and advanced analytics, the role of MSSQL in supporting these technologies becomes more significant. Predictive modeling, classification, and other advanced analytics require clean, consistent, and well-managed datasets. Preparing MSSQL infrastructure to handle such workflows should be a key area of focus. Research could examine how data cleansing and enrichment processes within MSSQL can be automated to streamline data preparation for machine learning applications.

Furthermore, integrating MSSQL with programming languages like R and Python could enable the efficient implementation of machine learning models directly within the database environment. This integration offers significant potential for predictive and prescriptive analytics, especially in industries like manufacturing, healthcare, and finance.

Dynamic optimization techniques also deserve further investigation. These methods, which adapt in real time to workload changes, could improve both query performance and resource management in MSSQL. For example, machine learning-based algorithms could monitor query execution patterns and dynamically optimize indexes, queries, and data structures. These advancements would ensure that MSSQL remains efficient even in high-demand environments with fluctuating workloads.

Finally, the role of MSSQL in federated learning scenarios should be explored. As data privacy and security become increasingly critical, federated learning offers a promising approach for distributed data management without sharing sensitive data. MSSQL's ability to manage large datasets while maintaining strong privacy controls makes it a suitable candidate for such architectures. Future research could focus on optimizing MSSQL for federated learning workflows, ensuring data privacy and security while enabling efficient data handling and processing. Additionally, its integration with data visualization tools like Power BI and Tableau could enhance reporting capabilities, enabling organizations to make data-driven decisions more effectively.

In conclusion, addressing these areas in future research will not only enhance the scalability, flexibility, and efficiency of MSSQL but also ensure its continued relevance in the evolving landscape of industrial data management.

Declaration

Ethics committee approval is not required.

References

[1] Malik, P. K., Sharma, R., Singh, R., Gehlot, A., Satapathy, S. C., Alnumay, W. S., Pelusi, D., Ghosh, U., & Nayak, J. (2021). Industrial internet of things and its applications in industry 4.0: *State of the art. Computer Communications,* 166, 125–139.

[2] Ghasemaghaei, M. (2021). Understanding the impact of big data on firm performance: The necessity of conceptually differentiating among big data characteristics. *International Journal of Information Management,* 57, 102055

[3] Fan, C., Yan, D., Xiao, F., Li, A., An, J., & Kang, X. (2021). Advanced data analytics for enhancing building performances: From data-driven to big data-driven approaches. *Building Simulation,* 14(1), 3–24.

[4] Naeem, M., Jamal, T., Diaz-Martinez, J., Butt, S. A., Montesano, N., Tariq, M. I., De-la Hoz-Franco, E., & De-La-Hoz-Valdiris, E. (2022). Trends and future perspective challenges in big data. *In Advances in Intelligent Data Analysis and Applications* (pp. 309–325). Springer.

[5] Ranjan, J., & Foropon, C. (2021). Big data analytics in building the competitive intelligence of organizations. *International Journal of Information Management,* 56, 102231.

[6] Larrea, M. L., & Urribarri, D. K. (2021). Visualization technique for comparison of time-based large data sets. *In Conference on Cloud Computing, Big Data & Emerging Topics* (pp. 179–187). Springer.

[7] Dinneen, J. D., & Brauner, C. (2017). Information-not-thing: Further problems with and alternatives to the belief that information is physical.

[8] Vaitis, M., Feidas, H., Symeonidis, P., Kopsachilis, V., Dalaperas, D., Koukourouvli, N., Simos, D., & Taskaris, S. (2019). Development of a spatial database and web-GIS for the climate of Greece. *Earth Science Informatics,* 12(1), 97–115.

[9] Amin, M., Romney, G. W., Dey, P., & Sinha, B. (2019). Teaching relational database normalization in an innovative way. *Journal of Computing Sciences in Colleges,* 35(2), 48–56.

[10] Alqithami, S. (2021). A serious-gamification blueprint towards a normalized attention. *Brain Informatics,* 8(1), 1–13.

[11] Oditis, I., Bicevska, Z., Bicevskis, J., & Karnitis, G. (2018). Implementation of NoSQL-based data warehouse. *Baltic Journal of Modern Computing,* 6(1), 45–55.

[12] Hrubaru, I., Talabă, G., & Fotache, M. (2019). A basic testbed for JSON data processing in SQL data servers. *In Proceedings of the 20th International Conference on Computer Systems and Technologies* (pp. 278–283).

[13] Chung, Y. G., Haldoupis, E., Bucior, B. J., Haranczyk, M., Lee, S., Zhang, H., Vogiatzis, K. D., Milisavljevic, M., Ling, S., Camp, J. S., et al. (2019). Advances, updates, and analytics for the computation-ready, experimental metal–organic framework database: Core MOF 2019. *Journal of Chemical & Engineering Data,* 64(12), 5985–5998.

[14] Bouros, P., & Mamoulis, N. (2019). Spatial joins: What's next? *SIGSPATIAL Special,* 11(1), 13–21.

[15] Myalapalli, V. K., Totakura, T. P., & Geloth, S. (2015). Augmenting database performance via SQL tuning. *In 2015 International Conference on Energy Systems and Applications* (pp. 13–18). IEEE.

[16] Pedrozo, W. G., & Vaz, M. S. M. G. (2014). A tool for automatic index selection in database management systems. *In 2014 International Symposium on Computer, Consumer and Control* (pp. 1061–1064). IEEE.

[17] Correia, J., Santos, M. Y., Costa, C., & Andrade, C. (2018). Fast online analytical processing for big data warehousing. *In 2018 International Conference on Intelligent Systems* (IS) (pp. 435–442). IEEE.

[18] Sulistiani, H., Setiawansyah, S., & Darwis, D. (2020). Penerapan metode agile untuk pengembangan online analytical processing (OLAP) pada data penjualan (Studi kasus: CV Adilia Lestari). *Jurnal CoreIT: Jurnal Hasil Penelitian Ilmu Komputer dan Teknologi Informasi,* 6(1), 50–56.

[19] Erdinç, H. N., Buluş, N., & Erdoğan, C. (2018). Veritabanı tasarımının yazılım performansına etkisi: Normalizasyona karşı denormalizasyon. *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 22(2), 887–895.

[20] Alshemaimri, B., Elmasri, R., Alsahfi, T., & Almotairi, M. (2021). A survey of problematic database code fragments in software systems. *Engineering Reports,* 3(10), e12441.

[21] Milicev, D. (2021). Hyper-relations: A model for denormalization of transactional relational databases. *IEEE Transactions on Knowledge and Data Engineering.*

[22] Chaparro-Cruz, I. N., & Montoya-Zegarra, J. A. (2021). Borde: Boundary and sub-region denormalization for semantic brain image synthesis. *In 2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)* (pp. 81–88). IEEE.

[23] Costa, R. L. D. C., Moreira, J., Pintor, P., dos Santos, V., & Lifschitz, S. (2021). A survey on data-driven performance tuning for big data analytics platforms. *Big Data Research,* 25, 100206.

[24] Chillón, A. H., Ruiz, D. S., & Molina, J. G. (2021). Towards a taxonomy of schema changes for NoSQL databases: The Orion language. *In International Conference on Conceptual Modeling* (pp. 176–185). Springer.

[25] Gupta, E., Sural, S., Vaidya, J., & Atluri, V. (2021). Attribute-based access control for NoSQL databases. *In Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy* (pp. 317–319).

[26] Santos, M. Y., Costa, C., Galvão, J., Andrade, C., Martinho, B. A., Lima, F. V., & Costa, E. (2017, July). Evaluating SQL-on-Hadoop for big data warehousing on not-so-good hardware. *In Proceedings of the 21st International Database Engineering & Applications Symposium* (pp. 242-252).

[27] Yang, F., Tschetter, E., Léauté, X., Ray, N., Merlino, G., & Ganguli, D. (2014, June). Druid: A real-time analytical data store. *In Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (pp. 157-168).

[28] Silva, Y. N., Almeida, I., & Queiroz, M. (2016, February). SQL: From traditional databases to big data. *In Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 413-418).

[29] Rahman, M. M., Islam, S., Kamruzzaman, M., & Joy, Z. H. (2024). Advanced Query Optimization in SQL Databases For Real-Time Big Data Analytics. *Academic Journal on Business Administration, Innovation & Sustainability*, 4(3), 1-14.

[30] Uzzaman, A., Jim, M. M. I., Nishat, N., & Nahar, J. (2024). Optimizing SQL databases for big data workloads: techniques and best practices. *Academic Journal on Business Administration, Innovation & Sustainability*, 4(3), 15-29.

[31] Pirzadeh, P., Carey, M., & Westmann, T. (2017, December). A performance study of big data analytics platforms. *In 2017 IEEE international conference on big data* (big data) (pp. 2911-2920). IEEE.

[32] Panwar, V. (2024). Optimizing Big Data Processing in SQL Server through Advanced Utilization of Stored Procedures. *Journal Homepage: http://www. ijmra. us*, 14(02).

[33] Ordonez, C. (2013, October). Can we analyze big data inside a DBMS?. *In Proceedings of the sixteenth international workshop on Data warehousing and OLAP* (pp. 85-92)

.