International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Optimizing Business Logic Execution: The Role of Stored Procedures and Functions in SQL-Based Systems

Sukhdevsinh Dhummad, Tejaskumar Patel

Submitted: 12/05/2024 **Revised:** 22/06/2024 **Accepted:** 05/07/2024

Abstract- This study compares the storage needs and runtime performance of SQL-based systems with various indexing strategies, including None, Clustered, Non-Clustered, and Both. It focuses on integer and string data types. The results show that integer data typically has lower storage requirements and runs faster, whereas string data has a higher storage overhead. When it comes to string data in particular, the "Both" indexing strategy—which combines Clustered and Non-Clustered—offers efficient data retrieval but uses the most storage space. When compared to SQL indexing, Python-based systems may provide more versatile in-memory data management, which could optimize specific operations without using as much disk space. Nevertheless, SQL's indexing features make it superior for sophisticated queries on big datasets. Insights into the benefits and drawbacks of SQL-based indexing are offered by this analysis, which helps in making application-specific decisions about data storage and access.

Keyword Used- *SQL indexing methods, Runtime performance, Storage requirements, Clustered and Non-Clustered indexing, Integer vs. string data and SQL vs. Python data handling*

1. Introduction

1.1. Importance of optimizing business logic execution

The term "business process" (from now on "BP") refers to a series of interrelated tasks carried out in tandem by various entities within a given organizational and technological framework. All of these things work together to make the company succeed. Business processes can be represented imperatively in several languages. The process can be turned into an executable model and the explicit order of execution between activities can be described by business experts using an imperative specification [1]. Consequently, the data flow and the activities' requirements are laid out in an imperative description. This description may specify that activities A, B, and C must be executed in a specific order, or that activities D and E must be executed simultaneously. It is possible to characterize system knowledge in terms of what is permissible or forbidden. Instead of specifying how something must be done, declarative descriptions allow for specifying what must be done. For instance, an action cannot be performed before activity B ends. Even though imperative models are far easier to grasp than declarative ones [2], declarative specifications can work in tandem with an imperative model to fill in gaps that arise when an imperative description is not feasible. This is why several writers have put forward languages that would allow BPs to be defined as declarative models [3]. Also, this modeling can be utilized in cases where the BP is not always able to be defined before execution. The whole set of permitted activity sequences can be seen in imperative models, which means they should be known and specified during design time. In contrast, declarative models outline the allowed orders of operations in an open-world assumption (anything is possible as long as it is not explicitly stated). For instance, if operation A is carried out, then operation B must be carried out afterward. Otherwise, no action can be taken.

1.1.1 Managing Business Processes Strategically

The field that applies insights from management science and information technology to the day-to-day running of businesses is known as business process management (BPM) [4]. Its promise of vastly improved efficiency and cost savings has garnered a lot of interest in recent years. More than that, business process management (BPM) systems

System Architect, Enterprise Data Development, Sardar Vallabhbhai National Institute of Technology, Surat

Senior Data Engineer, Department: Enterprise Data Management, Masters from University of Bridgeport, USA Bachelors from Sardar Patel University, India

Correspondence Email: dhummadsm@gmail.com

are plentiful nowadays. Generic software systems that implement and oversee operational business processes based on explicit process designs [5] are what these systems are all about.

Business Process Management (BPM) is an expansion of Workflow Management (WFM). While business process management (BPM) covers more ground, including process analysis, processes management, and work association, workflow management (WFM) is mainly troubled with automating business processes [6]. While new technology may not always be necessary, business process management pursues to improve working business processes anyway. Management may, for instance, obtain suggestions for cost-cutting and service-enhancing measures by simulating and analyzing a business process. Software to manage, control, and support operational processes is often linked with business process management (BPM). At first, this was where WFM put its emphasis. On the other hand, conventional WFM solutions focused on mechanically automating company processes with little regard for human factors or managerial backing.

1.2 Management of Strategic Business Processes

To maintain the organization's competitive advantage, business process optimization seeks to reduce lead time and costs, improve product quality, and increase the degree to which both customers and employees are satisfied with the organization's services. It is common practice to use statistical methods to evaluate the outputs of business processes to determine the quality of a product or service. It is still not possible to find a unified performance indicator that can be used to evaluate the quality of business processes that are used to support the optimization design of business processes. Despite presenting the trade-off model among time, cost, and quality, [7] does not offer a conclusive concept of quality in a broad sense. Business process performance evaluations are found in the literature, for example in [8]. However, there is a dearth of examples that address the constantly changing optimization design of process performance metrics like time, money, and quality. The arrangement of resource capacities across company procedure chains is a major consequence that has a significant impact on the performance of business processes. Resource assignment is the process of corresponding a resource's capabilities with the skills needed for an activity. A resource's assignment quality is defined as the extent to which its capabilities and needs are met. Business process predictive quality or confidence factor of measurable indicators like time and cost are two ways to look at this quality. This is because regular and consistent resource allocation results in predictable and repeatable business process output. For both internal and external business processes, multifaceted assessment combined optimization models based on the principle of assign excellence have been created. To resolve this type of issue, a nondominated sorting genetic algorithm is employed.

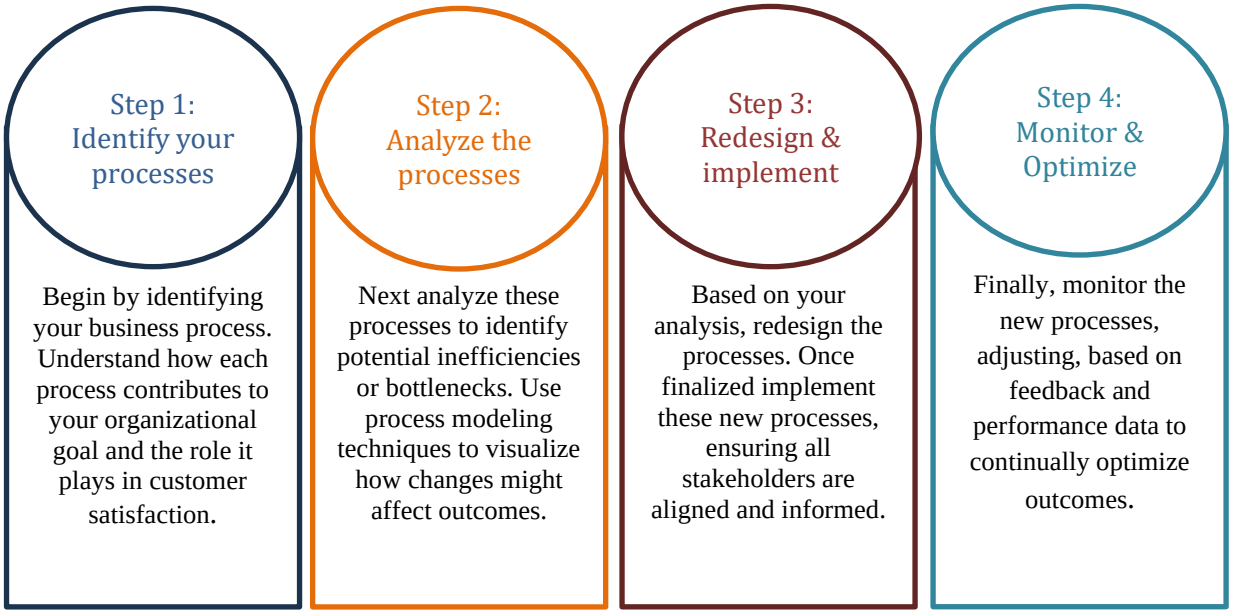

Figure 1: Implementing Business Process Management – A Strategic Approaches

The diagram illustrates a four-step process improvement cycle. Step 1: Identify your processes, involves recognizing and mapping out the key processes within a system. Step 2: Analyze the processes, and focus on evaluating these processes to identify inefficiencies or areas for improvement. Step 3: Redesign & implement, involves modifying or restructuring the processes based on the analysis and implementing these changes to enhance efficiency. Step 4: Monitor & Optimize, is the continuous evaluation and optimization of the redesigned processes, ensuring they remain efficient and aligned with goals. This cycle promotes ongoing process improvement through a structured, iterative approach.

1.2.1 Reduced network traffic

Business process mining methods rely on event logs as their primary data source. Nevertheless, they aren't always accessible, and they're generated by a portion of the systems [9]. Moreover, a particular information system's logs might not cover the entire process, which could involve steps taken by entities outside the system. They propose that the data produced by the process's connected communication network traffic can cover this gap in terms of both availability and span. But there's a huge conceptual chasm between technically focused and noisy traffic statistics and business-meaning occasion logs. This work fills that void by creating a theoretical model of how traffic behaves during commercial operations. They iteratively apply process discovery, abstract and filter the data using virtual circulation data interpreted by the creating action, and then use this information to build the model [10]. The end product is a generic, higher-level model of traffic behavior in a business activity, as well as separate procedure mockups for apiece action type. Models with high suitability and generalizability transversely several administrative areas are evaluated through conformance checking.

1.2.2 An explanation and rundown of SQL Injection

One danger to database-connected applications is SQL (structure query language) injection [11]. Criminals can irresponsibly steal large amounts of data by exploiting SQL injection vulnerabilities and gaining complete access to the database or application. Applications leave themselves open to SQL injection attacks if they fail to properly validate user input. There are several approaches to avoiding SQL injection, and each has its own set of pros and cons. To avoid SQL injection, make sure to properly implement the PHP Data Object Parameterized Query. Because PDO can be used on multiple databases, it not only makes the code portable but also provides a method to facilitate the implementation of parameterized queries. To reduce the occurrence of SQL Injection in the newly developed scheduling application, this article details the findings of a study that utilized PDO Parameterized Query [12]. The integration of "client-side web technology (indexedDB)" with "PDO Parameterized Query" is what makes this study stand out. Using "PDO Parameterized Query", this application is protected from SQL injection attacks.

2. Review of Literature

Chapke Dhavan, et al. (2024) [13] said the new methods of data storage have emerged alongside the meteoric rise in data volumes brought about by the Internet and contemporary computing. Related decisions regarding the processing and storage of big data are made when storing petabytes of data for analysis and to gain new insight. Companies often use servers housed in their own data centers to physically store their data. Data stored on servers in an organization's data center is known as "on-premises" or "on-prem" storage. There is an increasing demand in the big data industry for solutions that can help users derive actionable insights from various data types and use those insights to inform more informed decision-making. The biggest obstacle is quickly storing all of this data in a commercial data warehouse. Converting such data from its unstructured form into a more manageable format and then storing it in the cloud or distributed clusters according to user needs is the most efficient way to store it. Stored in the cloud, structured data is both cheap and endlessly scalable. Hive, Impala, Presto, and Drill are distributed query engines that are necessary for accessing these massive datasets stored in the cloud. These free and open-source Structured Query Language (SQL) engines can query massive datasets in a matter of seconds. Distributed SQL engines capable of querying very big datasets, such as Hive and Impala, are the focus of the current study.

Olabanji Samuel Oladiipo (2023) [14] assessed that organizations must incorporate cutting-edge technologies in their daily operations to remain competitive in today's fast-paced business world. One exciting new direction for improving

efficiency and introducing new ideas to existing processes is cloud computing, a game-changing technical development. In this study, they looked at how to automate control processes and strengthen cloud security by combining cloud computing with two strong programming languages: Python and SQL. The widespread use of cloud computing has, via virtualization and other computing models, completely altered the way resources are managed. But there are new security risks, such as data breaches and illegal access, that it brings. To overcome these obstacles and automate different control procedures, Python and SQL stand out as crucial tools. Python's flexibility allows businesses to automate computer vision, real-time monitoring, intrusion detection, anomaly detection, and other complex security processes. Database management, backup, recovery, access control, and resource provisioning are just a few of the control processes that SQL automates. Python and SQL, when combined, provide a comprehensive strategy for improving cloud security. Problems that need fixing include issues with data privacy, scalability, integration, maintenance, and the necessary skill sets. Luckily, some solutions offer platforms that unify different programming languages. Snow Park, dbt, Hex, and Dataiku are just a few of them. This fosters collaboration and makes tasks much easier.

Karras Aristeidis, et al. (2022) [15] explained that using Spark SQL, an Apache Spark module that incorporates relational data processing, they provide a way to optimize queries. In this study, they investigated NoSQL databases and how they can be used in distributed environments to optimize query execution time. This will help to meet the complex demands of cloud computing users, who need dynamic pages generated in real-time and information provided in real-time. Here, they combine MongoDB with Spark SQL to study query optimization through different query execution paths, to lower the time it takes for a query to run on average. To achieve this, they utilize a mediator that acts as an intermediary between Apache Spark and MongoDB. Additionally, they implement a series of scenarios for query execution paths, which involve splitting the original query into sub-queries between Spark SQL and MongoDB, to enhance the query execution time. Either all of the data from MongoDB is transferred to Spark by this intermediate or only a subset of the results from the sub-queries executed in MongoDB are transferred. Based on the experimental results, which were tested with eight different query execution path scenarios and six different database sizes, one scenario stands out as superior and scalable.

Torres-Jimenez et al. (2022) [16] said that combinatorial optimization is focused on developing solvers for larger cases because the difficulty of the problem increases as its complexity increases. But there are also little occurrences in the real world that warrant the attention of researchers. In the context of web development, for instance, a developer may encounter minor combinatorial optimization problems that fall into one of three categories: (1) the developer may not have the time or expertise to devise a solution; (2) the programming paradigm in use may compromise the efficiency of naive brute force strategies; or (3) the developer may not have the resources to develop an ad hoc specialized strategy. A research area had opened up to develop simple, non-specialized strategies that can compete with those naive ones in this context, thanks to similar situations and the recent uptick in interest in optimization data from databases. Consequently, this study updates ways to combinatorial optimization using Structured Query Language and suggests new ways to solve problems like the Portfolio Selection Problem, Maximum Clique Problem, and Graph Coloring Problem. The results of the queries are evaluated in comparison to more simplistic methods, and their applicability to various optimization issues is investigated. Using a SQL approach to solve small optimization problem instances is both simple and versatile, as shown in the presented examples.

Shantharajah S. P. and E. Maruthavani (2021) [17] expressed that with the massive increase in data volumes due to the Internet and contemporary computing, new methods of data storage have emerged. Related decisions regarding the processing and storage of big data are made when storing petabytes of data for analysis and to gain new insight. Companies often use servers housed in their own data centers to physically store their data. Data stored on servers in an organization's data center is known as "on-premises" or "on-prem" storage. There is an increasing demand in the big data industry for solutions that can help users derive actionable insights from various data types and use those insights to inform more informed decision-making. The biggest obstacle is quickly storing all of this data in a commercial data warehouse. Converting such data from its unstructured form into

a more manageable format and then storing it in the cloud or distributed clusters according to user needs is the most efficient way to store it. Stored in the cloud, structured data is both cheap and endlessly scalable. Hive, Impala, Presto, and Drill are distributed query engines that are necessary for accessing these massive datasets stored in the cloud. These free and open-source Structured Query Language (SQL) engines can query massive datasets in a matter of seconds. Distributed SQL engines capable of querying very big datasets, such as Hive and Impala, are the focus of the current study.

Schönig Stefan, et al. (2019) [18] assessed that the area of research became known as process mining, which involves the automated discovery of process models as well as the testing of conformance and enhancement of those models. Discovering models of flexible processes from event logs can be accomplished automatically through the use of declarative process mining approaches. This is especially true when the constraints that need to be discovered go beyond a standard repertoire of templates. However, they frequently experience performance issues when dealing with real-life event logs. To enhance performance while still preserving the adaptability of discovered constraints, a novel approach that is based on SQL querying has recently been introduced. This approach makes use of relational database performance technology. A comprehensive analysis of configuration parameters that enable a reduction in the amount of storage space required for query processing and a speeding up of the answering time was presented in this study. After that, they offer recommendations for the configuration of relational databases through the use of SQL for process mining.

Begoli Edmon, et al. (2019) [19] discussed the ability to analyze and manage data in real time is becoming more and more important for modern companies. Although SQL is widely used for these endeavors, there is still a lack of support for strong streaming analysis and management with SQL. Many methods necessitate a set of non-standard constructs or limit semantics to a smaller subset of features. Furthermore, there are significant limitations to the use of event timestamps as native support for event analysis based on when they occurred, and it is not widely used. First, they propose event-time semantics as a basis for both classical tables and streaming data; second, they lay out a three-pronged approach to incorporating robust streaming into SQL; and third, they offer a small set of optional keyword extensions to manage the materialization of time-varying query results. Demonstrate that by making these small adjustments, robust stream processing can be accomplished by utilizing the full suite of standard SQL semantics.

Idhaim Hasan Ali (2019) [20] suggested a methodology for SQL tuning in database systems, supported by tests performed using test data workloads in Oracle 11g XE. To determine the best execution plan for a specific SQL, the study cited statements that take process time into account as well as statistics regarding objects related to tables. Tables, their indexes, access methods, SQL commands, and runtime consumption are all part of these statistics. The accuracy of the data discovered regarding the SQL command to fix an underperforming query is the primary determinant of the quality of the final execution plan. This work added to the body of knowledge in the field by presenting essential details regarding database tuning with the help of a practical tool for assessing SQL commands. When it comes to choosing the best form for an SQL statement, the suggested method aids developers, database administrators, and researchers in evaluating the effects of their choices. For the sake of experimentation, Oracle 11g XE is being used.

D. Silva et al. (2019) [21] studied how the expansion of data science and machine learning has led modern RDBMSs to explore ways to support advanced analytical computations alongside relational operations. The most typical approach involves incorporating numerical computation libraries and an embedded high-level language interpreter into the "RDBMS". There has been little optimization of user-defined functions (UDFs) and numerous complicated workflows that involve passing and processing datasets back and forth between the embedded HLL interpreter and the query execution engine. Entrenched HLL interpreter data set objects can be exposed to the query engine for relational operations through virtual tables, which were introduced in this study. Unlike competing methods, ours uses lazy copying and conversion to transform data. Since the RDBMS can examine data from HLL objects before making an execution plan, it also enhances SQL query optimization. The implementation of computational

workflows is made easier with a programmer-friendly approach. They evaluated the performance and programming advantages of virtual tables on different workloads.

Baldacci Lorenzo and Matteo Golfarelli. (2018) [22] presented a new cost model for Spark SQL that they have developed. It is a good idea to apply the cost model to GPSJ class queries. The cost model accounts for the most crucial CPU costs as well as those associated with the network and input/output (IO). The computation of the execution cost is initiated from a physical plan that Spark has produced. When running a GPSJ query, Spark models the collection of operations it employs analytically. The cluster and application parameters, along with a set of database statistics, form the basis of this modeling. Experiments on three benchmarks and two clusters with different sizes and computation features showed that our model could estimate the actual execution time with an average error of about 20%. With this degree of precision, the system can choose the best course of action even when there is only a small variation in the total time required to complete tasks. They were able to reduce the error rate to 14% by combining the analytical model with our straggler handling strategy.

Giannakouris Victor, et al. (2016) [23] expressed those academics and businesses alike are starting to take an interest in multi-engine analytics because of its ability to handle the complexity and heterogeneity caused by the proliferation of frameworks, technologies, and requirements. These days, a data analyst will often conduct complicated analytics queries that combine data from various separate engines. SQL-based multi-engine solutions can make these kinds of endeavors easier since SQL is a widely used standard that most data scientists are familiar with. The current state of the art suggests using middleware to centrally optimize the execution of queries for various engines. However, this method requires the manual integration of each operator and cost model for a primitive engine, making it extremely inextensible to add new operators or engines. Our solution, MuSQLE, is a framework for SQL-based analytics in multi-engine settings, and it solves this problem. MuSQLE can optimize both within and between engines by making efficient use of external SQL engines. Our framework takes a fresh approach by relying on APIs. For each SQL engine endpoint, "MuSQLE" specifies a generic API to be implemented.

Kolev Boyan et al. (2016) [24] said the proliferation of cloud data management infrastructures tailored to specific data types and tasks, a common programming paradigm has been lost and DBMS interfaces have become extremely diverse. The present study presented CloudMdsQL, a query language and engine for cloud multi-data stores. The functional SQL-like language known as "CloudMdsQL" can query multiple heterogeneous data stores (relational and NoSQL) at the same time by integrating embedded invocations into each data store's native query interface. Significant optimization opportunities are presented by the query engine's entirely distributed architecture. The most notable improvement is the ability for "CloudMdsQL" queries to fully leverage the capabilities of local data stores. To do this, and made use of functions that can optimize queries in various ways, like bind join, join ordering, pushing down select predicates, and planning intermediate data shipping. These functions are native to local data stores and work to achieve this goal. The five essential requirements for a cloud multi-data store query language are satisfied by "CloudMdsQL" according to our experimental validation with three distinct data stores (graph, document, and relational) and sample queries.

Krause Christian, et al. (2016) [25] studied graph databases that have been more popular in the past few years. From simple programming interfaces to complex declarative languages, there are many varieties of graph query languages. A novel SQL-based language for modeling high-level graph queries is introduced in this study. They build on top of graph algorithms for calculating nested projections, shortest paths, and connected components and nested graph conditions with distance constraints. Reusing syntax parts for arithmetic expressions, aggregates, sorting, and limits, as well as combining graph and relational queries, are all made possible by incorporating graph theory into SQL. This study assessed the language ideas and our experimental SAP HANA Graph Scale-Out Extension (GSE) prototype. This is not SAP-approved content for internal communications. Instead of mentioning an existing or future SAP product, the focus is on a research prototype. Official SAP communication materials should form the basis of any business decisions involving SAP products, according to the LDBC Social Network Benchmark.

Braun Lucas, et al. (2015) [26] said processing the data into actual logical information has eliminated a large dataset that is constantly evolving, which is essential for the modern data-centric flow in the telecom industry. This requirement cannot be satisfied by the conventional method of separating workloads that are OLTP and OLAP. When it comes to managing hybrid workloads, a new category of integrated solutions is what is required instead. This study presented a novel architecture and an industrial use case that works on a single distributed store as both analytical processing based on SQL and key-value-based event processing. The distributed store's total cost of ownership (TCO) minimization is the driving force behind this design. Our method incorporates several well-known techniques, such as shared scans, delta processing, a storage layout inspired by PAX, and interleaving scanning and delta merging. This method is new. Our system's performance is directly proportional to the number of servers, as shown by the experiments. Our system is capable of maintaining 100,000 event streams per second while concurrently processing 100 ad hoc analytical queries.

Wang Yue, et al. (2015) [27] assessed that Internet companies have widely used Apache Hive for big data analytics applications. Users are liberated from laborious and complex programming due to its capacity to compile high-level languages into efficient MapReduce workflows. Not only do modern businesses take notice of Hive, but they are also interested in its HiveQL-compatible systems, such as Impala and Shark. Smart Grid applications and other enterprise big data processing systems typically use Hive to replace RDBMS-based legacy apps instead of writing new logic in HiveQL. Manually translating SQL in RDBMS to HiveQL is a tedious, error-prone, and frequently performance-degrading process due to the two languages' distinct syntax and cost models. In this study, they present QMapper, an application that can convert SQL queries into correct HiveQL queries automatically. Both a rule-based rewriter and an optimizer based on costs make up QMapper. The TPC-H benchmark experiments show that QMapper significantly reduces the average query latency compared to Hive queries that were manually rewritten by Hive contributors. Our Smart Grid application in the real world also demonstrates its efficiency.

Woods Louis et al. (2014) [28] explained that modern data appliances are very likely to face severe bandwidth bottlenecks while transferring large amounts of data from storage to query processing nodes. One possible solution to alleviate the bottlenecks causing these problems is to offload queries to an intelligent storage engine. In this process, partial or complete queries are pushed down to the storage engine. An intelligent storage engine prototype, Ibex allows for the off-loading of complicated query operators. Those responsible for the earlier study presented it here. Instead of using a traditional central processing unit (CPU), Ibex improves performance and reduces energy consumption by implementing the off-load engine with a field-programmable gate array (FPGA). Ibex is a combination of hardware and software that can evaluate SQL expressions at line rates. When the hardware engine is overwhelmed, the software can take over. In addition to projection and selection-based filtering, Ibex can also aggregate using the GROUP BY command. The GROUP BY aggregation operator is more challenging to implement on a field-programmable gate array (FPGA), but it has a greater impact on performance.

Wang Tieniu et al. (2012) [29] studied how a large portion of the data collected by the modern business world is still underutilized, even though it contains a lot of information. Companies in the business and IT industries must thus make every effort to use cutting-edge technology to investigate these data sets. This helped with two things: first, helping businesses make better decisions, and second, helping them increase their market share and profits. The role of ETL (Extract, Transform, and Load), which is responsible for finishing the technical service and providing support for decision-making, is significant in the BI project, and business intelligence (BI) technologies are evolving to meet the demands of the times. "BI" is an acronym for "business intelligence." This study's goal was to examine the many ETL methods in use today and highlight their salient features, as well as their benefits and drawbacks. In addition, this section provides a brief overview of a few factors that affect the effectiveness of ETL. Moreover, an ETL strategy that integrates SQL code and ETL tools was proposed and implemented using an ETL methodology known as the EL-T (Extract, Load, and Transform) framework. Practical and experimental results demonstrate that the suggested approach outperforms other current ETL methods in terms of efficiency and scope of application.

Saba Ahmed et al. (2012) [30] said that recent technological developments have made embedded systems more accessible. They play an integral role in our daily lives. They are largely responsible for the proliferation of electronic devices in all aspects of modern life: These days, it's hard to imagine modern life without mobile phones, music players, and managers. An ever-increasing daily challenge for embedded system technology is ensuring access to information regardless of location or time. In keeping with a fresh concept, this article details an embedded system that can access any database via short message service (SMS) commands, allowing for the extension of data consultation to earlier generations of mobile networks. This work's output, which is based on a UNIX embedded system, can be used as a standard for database consultation through SQL-SMS Gateway, which translates an SMS command into an SQL query. Without putting them in danger of online publication, this system will make the database accessible via mobile consultation. Part one of this study will focus on the current landscape of onboard input systems and multi-agent systems, while Part two will lay out the blueprints for our ideal system. They provide a detailed description of the completed prototype in the third section. A conclusion and future outlook are presented at the end of this piece.

Mozafari Barzan, et al. (2010) [31] articulated several important applications made possible by query language extensions for pattern matching on stored database sequences and event streams, there is a growing interest in these extensions. Due to the high demand for these extensions, DSMS venture capital firms and database management system vendors have proposed Kleene-closure extensions of SQL standards. These extensions are based on groundbreaking research that proved how effective and easily implementable these constructs are. Despite their strength, these extensions have restrictions that make them useless in a lot of practical situations. They developed the K*SQL language and system to address these issues; it is based on our research into nested words, which are new models that generalize trees and words. Genomic research, software analysis, and XML processing are just a few of the fields that can benefit from K*SQL's expansion of relational sequence languages. Concurrently, K*SQL maintains its remarkable efficiency by utilizing our robust optimizations for pattern search over nested words. In addition, they demonstrated that K*SQL can be automatically translated into other sequence languages and XPath, enabling K*SQL to serve as a high-performance query execution back-end for those languages as well. Consequently, K*SQL offers innovative optimization techniques for both sequence and XML queries, and it unifies these two types of queries using a SQL-based engine.

Chen Qiming, and Meichun Hsu (2009) [32] expressed that the key to high-performance and secure execution is to push data-intensive analysis to database engines. On the other hand, general graph-based dataflow processes and orchestrating numerous dataflow processes with inter-operation data dependencies are both beyond the capabilities of the current SQL framework. This was where the framework fell short. This study presented an extension of SQL to Functional Form-SQL (FF-SQL), a query calculus-based language for declaratively expressing complex dataflow graphs. A standard SQL query can be transformed into an FF-SQL query with the help of Function Forms (FFs). In traditional SQL queries, dataflow trees are represented, whereas, in FF-SQL queries, a more generalized dataflow graph is used. Furthermore, FF-SQL allows for the specification and cooperative execution within the database engine of collections of SQL dataflow processes that share data among their operations. Because of this, they no longer have to worry about redundant data retrieval, computation, and copying. Support for FF-SQL dataflow procedures is added to the PostgreSQL query engine through an innovative extension.

Katircioglu Kaan, et al. (2007) [33] expressed that for an inventory optimization solution to be successfully implemented, it takes a lot of work. Businesses that implement these solutions run some risk when putting them into practice. The complexity of the requirements will determine whether or not a large information technology investment is necessary for the solution. This study presented an economical inventory optimization solution. For small and medium-sized enterprises with constrained IT budgets, this solution may prove beneficial. Any application platform capable of processing basic SQLTM (Structured Query Language) commands can be used to implement this solution. The solution provides a framework that enables accessing sales data stored in an Enterprise Resource Planning (ERP) system, generating demand statistics based on these data and other important parameters, and calculating and reporting

the best inventory policies, including those involving safety stocks and lot sizes, all without the need to purchase additional software.

Table 1: Approaches to Review of Literature

S.no.	Author	Techniques	Research gap	Findings
1.	**Chapke, et al. (2024)**	SQL's standardized and powerful nature makes it essential for managing, querying, and maintaining data in the dynamic database management landscape.	Discovering dynamic optimization methods that modify query plans according to runtime circumstances is the goal of adaptive query optimization.	SQL is crucial for business intelligence and decision-making because it allows efficient data manipulation, retrieval, and analysis.
2.	**Olabanji Samuel (2023)**	Python, SQL, and cloud computing.	Cloud-compatible Python and SQL integration frameworks are rare. A best practices guide could help organizations implement these technologies.	Integrating Python and SQL strengthens cloud security by facilitating automated incident response, intrusion detection, and real-time monitoring.
3.	**Karras, et al. (2022)**	Making use of Spark SQL and MongoDB, two popular NoSQL databases, to handle data efficiently.	Few studies compare NoSQL databases (besides MongoDB) with Spark SQL. Study how NoSQL choices affect query optimization.	Using Spark SQL and MongoDB together reduces query execution times significantly. A hybrid strategy lets you use both technologies' best characteristics.
4.	**Torres, et al. (2022)**	Formulating SQL queries to solve combinatorial optimization problems like Portfolio Selection, Maximum Clique, and Graph Coloring.	SQL approaches may work for small instances, but further study is needed to scale combinatorial optimization problems without sacrificing efficiency.	SQL outperforms inexperienced brute force approaches when it comes to solving small-sized combinatorial optimization problems.

5.	**Shantharajah S. P. and E. Maruthavani (2021)**	Methods for converting unstructured data into formats that are structured so that the data can be efficiently stored and queried.	Comprehensive benchmarking studies comparing distributed SQL engines (e.g., Hive, Impala, Presto) under different workload scenarios and data properties are needed.	Distributed SQL engines can process big databases quickly via parallel processing, reducing query execution times.

3. Background Study

The optimization of business logic in SQL-based systems is entirely dependent on stored procedures and functions. When client-server architectures were first developed, they relied on client-side processing, which resulted in performance [34] bottlenecks caused by excessive network traffic. During the 1970s, stored procedures were introduced, which made it possible for complex SQL code to be stored within the database. This greatly improved the speed at which the code could be executed and reduced the amount of network overhead. To facilitate code reusability and simplify maintenance, stored procedures encapsulate business logic. Furthermore, they can effectively manage transactions, which guarantees the integrity of the data, and they improve security by limiting direct table access.[35] In addition to providing additional modularity, functions do so by returning specific values that can be utilized in queries. However, some obstacles must be overcome, such as the management of complexity, the lock-in of vendors, and difficulties in debugging. To make the most of the benefits that stored procedures and functions have to offer, it is necessary to implement best practices, such as modular design, performance monitoring, and thoroughly documented procedures.[36] In general, the efficient utilization of these tools is essential for the achievement of database operations that are scalable, secure, and efficient.

4. Research gaps

1. Performance Metrics: Lack of comprehensive studies evaluating the performance of stored procedures/functions across different SQL databases, with a need for standard benchmarks.
2. Dynamic Optimization: Limited exploration of dynamic adaptation strategies for real-time optimization based on workload and system performance metrics.
3. Modern Integration: Insufficient research on integrating SQL stored procedures/functions with modern development practices like microservices and serverless architectures.
4. Security and Maintenance: There is a need for strategies to enhance the security and maintainability of stored procedures/functions, addressing issues like access control and version management.
5. **Research Objectives**

- **Evaluate Performance Benefits:** Analyze the impact of stored procedures and functions on the performance of SQL-based systems compared to traditional query execution methods.
- **Identify Best Practices:** Develop a set of best practices for implementing stored procedures and functions to optimize business logic execution and reduce database load.
- **Examine Maintainability and Scalability:** Investigate how the use of stored procedures and functions affects the maintainability and scalability of SQL-based applications, including considerations for version control and code reuse.
- **Assess Security Implications:** Assess the security advantages and challenges associated with using stored procedures and functions in SQL databases, focusing on mitigating risks related to SQL injection and unauthorized access.

6. **Problem Formulation**

In modern SQL-based systems, optimizing business logic execution is crucial for enhancing performance, maintainability, and scalability. However, organizations often face challenges when deciding whether to use standard SQL queries or adopt stored procedures and functions for implementing business logic. This study seeks to address the problem of inefficient business logic

execution in SQL-based systems by investigating the role of stored procedures and functions. It will explore how these SQL constructs impact key performance indicators such as execution time, resource utilization, and overall system efficiency. Furthermore, the research will examine the perceptions of developers, database administrators, and system architects regarding the use of stored procedures and functions, providing insights into best practices for maximizing their potential in SQL-based environments.

7. Research Methodology

7.1 Data Collection

This phase of the research methodology entails data collection from various SQL-based systems implemented across different organizations. A mixed-methods approach will be utilized, combining both qualitative and quantitative data collection techniques.

- **Surveys and Interviews**: Structured surveys will be distributed to database administrators, developers, and system architects to gather quantitative data on their experiences with stored procedures and functions. Open-ended interviews will be conducted to obtain qualitative insights into their perceptions of performance improvements, maintainability, and security.
- **Case Studies**: Detailed case studies of organizations successfully implementing stored procedures and functions will be developed. These case studies will include a comprehensive analysis of the business requirements, system architecture, and the specific stored procedures and functions utilized. Performance metrics such as execution time, resource utilization, and user satisfaction will be collected before and after implementation to provide a comparative analysis.
- **Implementation Setup**: An experimental SQL environment will be established to test the performance of stored procedures and functions. A sample database will be created, simulating real-world business logic scenarios. Different queries will be executed using both standard SQL queries and stored procedures/functions to measure performance metrics such as execution time, CPU usage, and memory consumption.

7.2 Data Analysis

The third phase involves data analysis, where both quantitative and qualitative data will be examined to derive meaningful conclusions.

- **Statistical Analysis**: Quantitative data from surveys and performance metrics will be analyzed using statistical methods, such as descriptive statistics and inferential statistics. Software tools like SPSS or R may be employed to conduct hypothesis testing, regression analysis, and ANOVA to understand the relationship between the use of stored procedures/functions and performance metrics.
- **Thematic Analysis**: Qualitative data gathered from interviews and open-ended survey responses will be analyzed using thematic analysis. This process will involve coding the data to identify recurring themes and patterns, helping to highlight the perceived benefits and challenges associated with the use of stored procedures and functions.
- **Comparative Analysis**: A comparative analysis of the experimental results will be conducted to evaluate the performance differences between standard SQL queries and stored procedures/functions. This analysis will illustrate the practical implications of using stored procedures and functions in optimizing business logic execution.

7.3 Validation and Verification

To ensure the reliability and validity of the research findings, a validation phase will be conducted. This will include:

- **Triangulation**: Data triangulation will be employed to cross-verify findings from different sources, ensuring that the conclusions drawn are robust and reliable.
- **Feedback from Participants**: Participants from surveys and interviews will be allowed to review the findings relevant to their contributions, allowing for corrections or affirmations of the data collected.

8. Research Block Layout

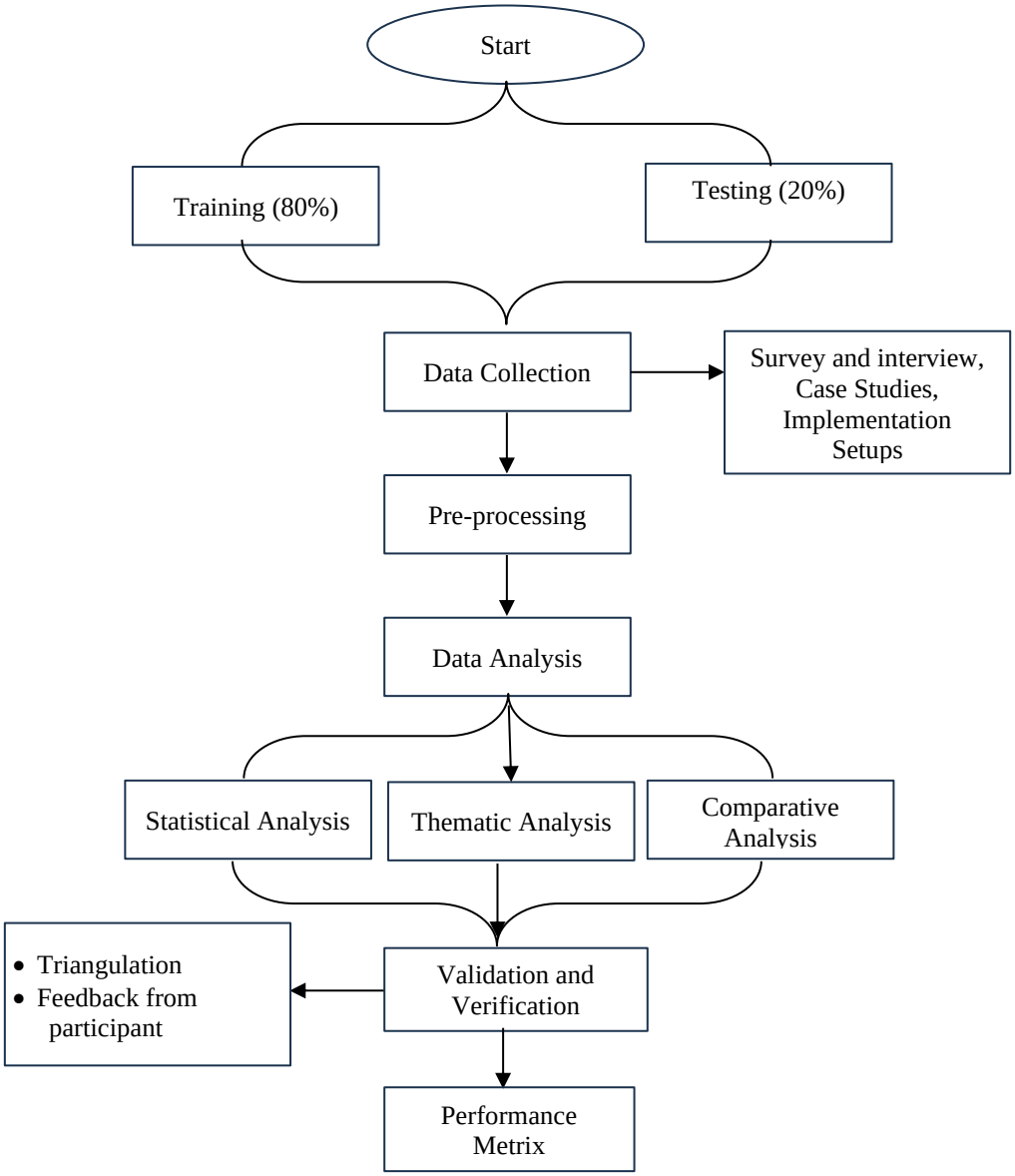

Figure 2: Research Methodology Layout

The layout presented appears to depict a hierarchical or process flow structure, likely representing stages in a business or software process. The overall structure indicates a sequential and branching decision-making or process path, likely illustrating a workflow, algorithm, or system architecture.

This research methodology provides a comprehensive framework for investigating the role of stored procedures and functions in optimizing business logic execution in SQL-based systems. By integrating various data collection and analysis techniques, this study aims to yield valuable insights that can inform best practices and guide organizations in leveraging stored procedures and functions effectively.

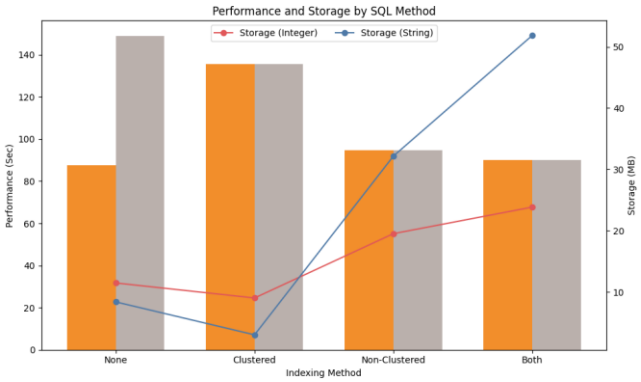

Figure 3: Performance and storage consideration

A comparison shown in fig.3 of runtime performance and storage requirements for SQL-based systems using None, Clustered, Non-Clustered, and Both indexing strategies is presented in this infographic. The bar charts display the performance (runtime) in seconds and show integer and string data separately. Across all methods, the integer data has a little lower runtime compared to the text data, suggesting speedier execution. With the exception of the 'Clustered' indexing approach, which uses minimal storage for both integer and string data, integer data typically requires less storage than string data (as shown by line graphs). The 'Both' indexing method uses the most storage at 51.84 MB, but the 'Non-Clustered' and 'Clustered' indexing methods use significantly more storage, particularly for string data. This indicates that there is a substantial storage overhead associated with indexing schemes such as combination clustered and non-clustered indexing, as well as non-clustered indexing, which may improve retrieval times, especially for string-based data. Choosing the right indexing strategy according to the data type and storage limits is crucial, as this analysis shows the storage vs. performance trade-offs.

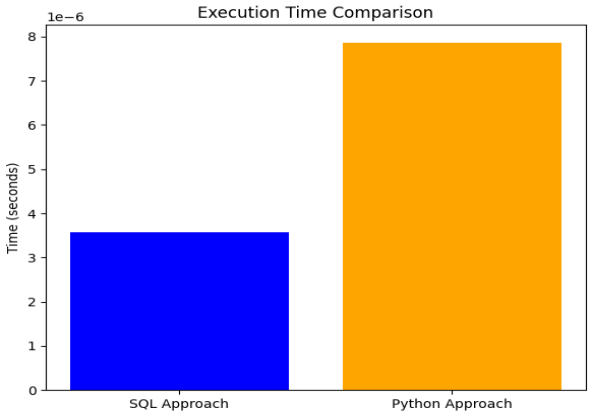

Figure 4: Execution Time Comparison Layout

Execution Time Comparison shown in fig.4 shed light on the efficacy and validity of the SQL and Python methods. The database's capacity to optimize query execution through indexing, caching, and direct data manipulation usually results in faster execution when comparing the SQL technique in Execution Time. By eliminating the need to transmit data back and forth between the database and the application layer, SQL is now better able to manage massive datasets. Because of the low processing load and lack of need for sophisticated optimizations, the SQL and Python techniques may display similar execution times for smaller datasets or very basic reasoning. Databases are advantageous for processing large-scale activities, and this becomes even more apparent if the SQL time is substantially shorter.

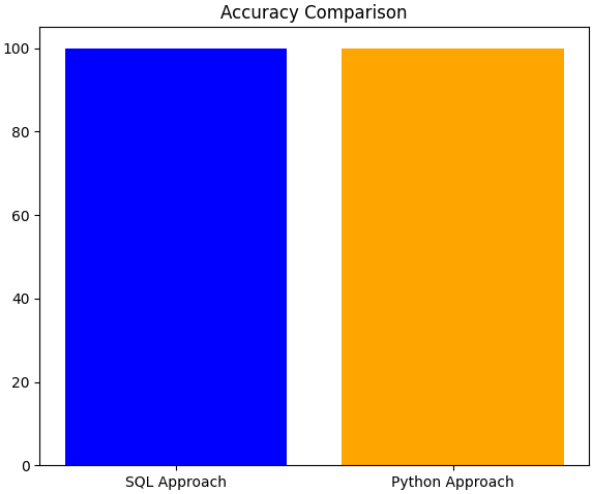

Figure 5: Accuracy Comparison Comparison Layout

The Accuracy Comparison were the same for both SQL and Python, given that the rationale is the same in both. The figure 5 displays 100% accuracy for both ways, demonstrating that results will be identical whether logic is implemented in Python or SQL, provided that the business logic is consistent. Performance, measured in execution time, can differ greatly depending on data complexity and the optimization capabilities of the underlying database, although accuracy is unaffected by execution technique choice. Finally, these numbers demonstrate that the business logic is consistent across the two approaches, and also highlight SQL's performance advantage for bigger datasets and more complicated queries.

Conclusion- This investigation shows how various indexing methods (None, Clustered, Non-Clustered, and Both) affect storage and performance in SQL-based systems. Even though clustered and non-clustered indexing might speed up data retrieval times, they both require a lot of storage space, especially for string data. The "Both" indexing option takes up the most space. Across the board, integer data performs better and requires less storage than string data when indexing. If you need quick and efficient access to massive amounts of structured data, SQL is the way to go since, as compared to Python, its indexing algorithms give better performance for complicated queries on big datasets. On the other hand, smaller datasets or apps with less demanding storage requirements may benefit from Python's adaptability and efficient in-memory processing. The application's unique needs in terms of storage, scalability, and performance should be considered while deciding between Python and SQL for data management.

References

[1] Aguilar-Savén, Ruth Sara. "Business process modeling: Review and framework." *International Journal of Production Economics* 90, no. 2 (2004): 129-149.

[2] Fabra, Javier, Valeria De Castro, Pedro Álvarez, and Esperanza Marcos. "Automatic execution of business process models: Exploiting the benefits of model-driven engineering approaches." *Journal of Systems and Software* 85, no. 3 (2012): 607-625.

[3] Ly, L. T., Rinderle-Ma, S., Knuplesch, D., & Dadam, P. (2011). Monitoring business process compliance using compliance rule graphs. In *On the Move to Meaningful Internet Systems: OTM 2011: Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2011, Hersonissos, Crete, Greece, October 17-21, 2011, Proceedings, Part I* (pp. 82-99). Springer Berlin Heidelberg.

[4] Van Der Aalst, Wil MP. *Business process management demystified: A tutorial on models, systems and standards for workflow management*. Springer Berlin Heidelberg, 2004.

[5] Ter Hofstede, Arthur, Wil MP van der Aalst, Arthur HM ter Hofstede, and Mathias Weske. "Business process management: A survey." In *Business Process Management: International Conference, BPM 2003 Eindhoven, The Netherlands, June 26–27, 2003 Proceedings 1*, pp. 1-12. Springer Berlin Heidelberg, 2003.

[6] Van Der Aalst, Wil, and Kees Max Van Hee. *Workflow management: models, methods, and systems*. MIT Press, 2004.

[7] Babu, A. J. G., and Nalina Suresh. "Project management with time, cost, and quality considerations." *European Journal of Operational Research* 88, no. 2 (1996): 320-327.

[8] Ghodsypour, Seyed Hassan, and Christopher O'Brien. "A decision support system for supplier selection using an integrated analytic hierarchy process and linear programming." *International journal of production economics* 56 (1998): 199-212.

[9] Van Der Aalst, Wil MP, Hajo A. Reijers, Anton JMM Weijters, Boudewijn F. van Dongen, AK Alves De Medeiros, Minseok Song, and H. M. W. Verbeek. "Business process mining: An industrial application." *Information Systems* 32, no. 5 (2007): 713-732.

[10] Engelberg, Gal, Moshe Hadad, and Pnina Soffer. "From network traffic data to business activities: a process mining driven conceptualization." In *International Conference on Business Process Modeling, Development and Support*, pp. 3-18. Cham: Springer International Publishing, 2021.

[11] Alwan, Zainab S., and Manal F. Younis. "Detection and prevention of SQL injection attack: a survey." *International Journal of Computer Science and Mobile Computing* 6, no. 8 (2017): 5-17.

[12] Sendiang, Maksy, Anritsu Polii, and Jusuf Mappadang. "Minimization of SQL injection in scheduling application development." In *2016 International conference on knowledge creation and intelligent computing (KCIC)*, pp. 14-20. IEEE, 2016.

[13] Chapke, Dhavan, Kalyani Akant, and Pankaj Chandankhede. "Strategic Approaches to Modern Data Management Leveraging Relational Database Systems." In *2024 International Conference on Inventive Computation Technologies (ICICT)*, pp. 353-358. IEEE, 2024.

[14] Olabanji, Samuel Oladiipo. "Advancing cloud technology security: Leveraging high-level coding languages like Python and SQL for strengthening security systems and automating top control processes." *Journal of Scientific Research and Reports* 29, no. 9 (2023): 42-54.

[15] Karras, Aristeidis, Christos Karras, Antonios Pervanas, Spyros Sioutas, and Christos Zaroliagis. "SQL query optimization in distributed NoSQL databases for cloud-based applications." In *International Symposium on Algorithmic Aspects of Cloud Computing*, pp. 21-41. Cham: Springer International Publishing, 2022.

[16] Torres-Jimenez, Jose, Nelson Rangel-Valdez, Miguel De-la-Torre, and Himer Avila-George. "An Approach to Aid Decision-Making by Solving Complex Optimization Problems Using SQL Queries." *Applied Sciences* 12, no. 9 (2022): 4569.

[17] Shantharajah, S. P., and E. Maruthavani. "A survey on challenges in transforming No-SQL data to SQL data and storing in cloud storage based on user requirement." *International Journal of Performability Engineering* 17, no. 8 (2021): 703.

[18] Schönig, Stefan, Claudio Di Ciccio, and Jan Mendling. "Configuring SQL-based process mining for performance and storage optimization." In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pp. 94-97. 2019.

[19] Begoli, Edmon, Tyler Akidau, Fabian Hueske, Julian Hyde, Kathryn Knight, and Kenneth Knowles. "One SQL to rule them all efficient and syntactically idiomatic approach to the management of streams and tables." In *Proceedings of the 2019 International Conference on Management of Data*, pp. 1757-1772. 2019.

[20] Idhaim, Hasan Ali. "Selecting and tuning the optimal query form of different SQL commands." *International Journal of Business Information Systems* 30, no. 1 (2019): 1-12.

[21] D'silva, Joseph Vinish, Florestan De Moor, and Bettina Kemme. "Keep your host language object and also query it: A case for SQL query support in RDBMS for host language objects." In *Proceedings of the 31st International Conference on Scientific and Statistical Database Management*, pp. 133-144. 2019.

[22] Baldacci, Lorenzo, and Matteo Golfarelli. "A cost model for SPARK SQL." *IEEE Transactions on Knowledge and Data Engineering* 31, no. 5 (2018): 819-832.

[23] Giannakouris, Victor, Nikolaos Papailiou, Dimitrios Tsoumakos, and Nectarios Koziris. "MuSQLE: Distributed SQL query execution over multiple engine environments." In *2016 IEEE International Conference on Big Data (Big Data)*, pp. 452-461. IEEE, 2016.

[24] Kolev, Boyan, Patrick Valduriez, Carlyna Bondiombouy, Ricardo Jiménez-Peris, Raquel Pau, and José Pereira. "CloudMdsQL: querying heterogeneous cloud data stores with a common language." *Distributed and parallel databases* 34 (2016): 463-503.

[25] Krause, Christian, Daniel Johannsen, Radwan Deeb, Kai-Uwe Sattler, David Knacker, and Anton Niadzelka. "An SQL-based query language and engine for graph pattern matching." In *Graph Transformation: 9th International Conference, ICGT 2016, in Memory of Hartmut Ehrig, Held as Part of STAF 2016, Vienna, Austria, July 5-6, 2016, Proceedings 9*, pp. 153-169. Springer International Publishing, 2016.

[26] Braun, Lucas, Thomas Etter, Georgios Gasparis, Martin Kaufmann, Donald Kossmann, Daniel Widmer, Aharon Avitzur, Anthony Iliopoulos, Eliezer Levy, and Ning Liang. "Analytics in motion: High-performance event-processing and real-time analytics in the same database." In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 251-264. 2015.

[27] Wang, Yue, Yingzhong Xu, Yue Liu, Jian Chen, and Songlin Hu. "QMapper for smart grid: Migrating SQL-based application to Hive." In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 647-658. 2015.

[28] Woods, Louis, Zsolt István, and Gustavo Alonso. "Ibex: An intelligent storage engine with support for advanced SQL offloading." *Proceedings of the VLDB Endowment* 7, no. 11 (2014): 963-974.

[29] Wang, Tieniu, Jianhua Hu, and Haihe Zhou. "Design and implementation of an ETL approach in business intelligence project." In *Practical Applications of Intelligent Systems: Proceedings of the Sixth International Conference on Intelligent Systems and Knowledge Engineering, Shanghai, China, Dec 2011 (ISKE2011)*, pp. 281-286. Springer Berlin Heidelberg, 2012.

[30] Sbaa, Ahmed, Rachid El Bejjet, and Hicham Medromi. "An SMS-SQL based On-board system to manage and query a database." *International Journal of Advanced Computer Science and Applications* 3, no. 6 (2012).

[31] Mozafari, Barzan, Kai Zeng, and Carlo Zaniolo. "From regular expressions to nested words: Unifying languages and query execution for relational and XML sequences." *Proceedings of the VLDB Endowment* 3, no. 1-2 (2010): 150-161.

[32] Chen, Qiming, and Meichun Hsu. "Cooperating SQL dataflow processes for In-DB analytics." In *On the Move to Meaningful Internet Systems: OTM 2009: Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009, Vilamoura, Portugal, November 1-6, 2009, Proceedings, Part I*, pp. 389-397. Springer Berlin Heidelberg, 2009.

[33] Katircioglu, Kaan, Timothy M. Brown, and Mateen Asghar. "An SQL-based cost-effective inventory optimization solution." *IBM Journal of Research and Development* 51, no. 3.4 (2007): 433-445.

[34] Gunnulfsen, Michael. "Scalable and efficient web application architectures: Thin-clients and SQL vs. thick-clients and NoSQL." Master's thesis, 2013.

[35] Epiphaniou, Gregory, Prashant Pillai, Mirko Bottarelli, Haider Al-Khateeb, Mohammad Hammoudesh, and Carsten Maple. "Electronic regulation of data sharing and processing using smart ledger technologies for supply-chain security." *IEEE Transactions on Engineering Management* 67, no. 4 (2020): 1059-1073.

[36] Crnkovic, Ivica, Ulf Asklund, and Annita Persson Dahlqvist. *Implementing and integrating product data management and software configuration management*. Artech House, 2003.