Harsha Vardhan Reddy Kavuluri
WISSEN Infotech INC, United States

Received: 13-04-2021; Revised: 27-05-2021; Accepted: 17-06-2021

Automating Oracle Database Performance Tuning Through AI-Based Workload Analysis

Abstract
Oracle databases effectiveness in enterprise applications is significant. However, the tuning process is time-consuming, and it requires profound knowledge. Static tuning strategies employed in previous times are inadequate when it comes to working with fluctuating loads and changing application topologies. This paper proposes a well-articulated plan for Oracle database performance tuning using AI workloads auto evaluation. This proactive identification is achieved through performance prediction using Machine Learning (ML) algorithms, the characterization of workload, and performance history data from the baseline analysis. This paper focuses on system formulation, data accumulation, feature extraction, model development, and decision-making processes. Overall, the effectiveness of the proposed framework was thoroughly evaluated using real clustering enterprise workload, resulting in significant achievements in query execution time, resource consumption, and system instructiveness. This paper also outlines how the system can be customized on how it works depending on the workload placed on it and its performance compared to that of the rule-based tuning method. The experiment's findings prove the proposed idea of the AI-based tuning approach providing performance improvement while simultaneously eliminating administrative burden. This is where the proposed solution shows that it is possible to overcome the increased database complexity and its requirements for high availability, scalability, and performance through intelligent automation.
Keywords: Oracle database tuning, workload analysis, machine learning, query optimization, AI.

1. Introduction

1.1. Role of Automating Oracle Database Performance Tuning

As it applies to the performance tuning and high availability of Oracle databases, database automation is important to minimize overhead. As the databases grow and continually change, manual tuning is too time-consuming and incorrect, and it cannot adapt to the changing conditions of business workloads. [1-4] Automating processes in an organization has several advantages. They include: First, an organization can experience better performance; Second, the cost reduction; Third, the improvement of user experience. Several factors relate to the role of automation in the Oracle database performance tuning, as stated next:

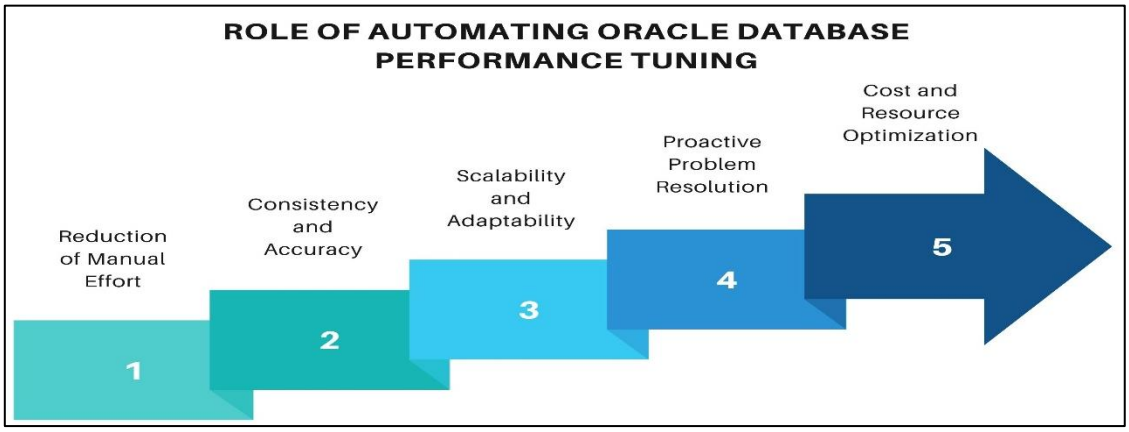

Figure 1: Role of Automating Oracle Database Performance Tuning

• Reduction of Manual Effort: Automation has many benefits in Oracle Database performance tuning, one of which is that it greatly reduces the workload of DBAs. Historically, DBAs were expected to review the performance data themselves, diagnose the system bottlenecks, and make tuning decisions on what needs to be done, be it indexes, queries, or system parameters. This procedure is quite tiresome and may cause several errors due to its manual nature, especially involving an extensive database. It automates these tasks in order to help the system monitor and tune the databases routinely without the involvement of DBAs. Therefore, other more important duties likely to be handled by DBAs include designing the architecture or planning for the system's scalability.

• Consistency and Accuracy: A top advantage of performing Oracle database performance tuning by automation is that the tuning is more consistent and accurate. The system administrators usually tune up the RDBMS. It tends to be inconsistent if done by several DBAs and is inconsequential if done by a single DBA because the process is based on the DBA’s expertise and experience. Automated systems are more prescribed by setting programs and parameters with machine learning, which considers performance data quantitatively. These systems can apply optimum tuning actions according to the runtime statistics, and it eliminates the uncertainty of interpreting the performance changes by other workers. This, in turn, leads to more accurate and reliable changes in settings that will guarantee better

and enhanced database performance.

• Scalability and Adaptability: With more and more databases being used in organizations, both in their capacity and in terms of functionality, it is crucial to determine how well they can handle increasing workloads. Systems that can automatically tune also grow with the database, which is best for minimizing bad hotspots and other volumes, interruptions, or fluctuations. Since no human intervention is required to automate tuning, variations in workload are easier to address. For example, if the number of users or type of query changes, it can identify that and change the allocation of CPU, memory, and indexer settings. This flexibility gives the system the capability to accommodate change that is both expected and unexpected without any effect on the DB.

• Proactive Problem Resolution: Automated performance tuning also features the ability to address a number of potential issues before they cause issues for a system. Thus, using machine learning and predictive methods, computerized systems can identify potential problems, for example, long query execution time, delays in disk input/output operations, or CPU overloads based on preceding performance. Such problems can be solved if detected on time to avoid becoming complex issues that may affect many users. For instance, it may suggest particular indexes to create or alter memory settings in order to keep queries from slowing down as the amount of data grows. This addresses the long-term database performance issue and ensures the key service quality of reliable performance timestamp processing with minimal disruption.

• Cost and Resource Optimization: It is also an advantage that the process of database performance tuning is automatable and, thus, will save considerable costs and resources. As database resources continue to be in high demand, especially in cloud-based systems, managing the system resources becomes essential. There are self-sufficient systems that keep track of the usage of resources such as the CPU, memory, and disk usage. While using the work as a guideline, resource utilisation has benefits; it helps avoid over-provisioning, which leads to wastage of resources to running costs, and under-provisioning, which leads to a decline in performance. In addition, automation enables the discovery of inefficiencies in the use of resources by suggesting improvements such as query optimization, better indexing, and optimisation of resource usage, among others. This also helps to cut operational costs while at the same time ensuring that the database can readily deal with all the necessary work during some busy periods without requiring additional hardware or other infrastructural overhaul.

1.2. Leveraging AI-Based Workload Analysis

The integration of AI for workload analysis has brought a new sense of intelligence to how databases administer the performance of the workloads, which is in contrast to a generic one for handling various database workloads. Typically, performance tuning involves setting human intervention or a set of heuristics to correct performance problems, which is inadequate when it comes to the complex and dynamic nature of the current DB systems. In contrast, AI-based workload analysis automatically

identifies and, over time, develops algorithms and machine learning models to understand the underlying conditions and patterns, allowing it to adapt to changes in the workload on the go. [5,6] In essence, the assessment of workloads using artificial intelligence is designed to determine various specifics related to the type of queries run, the amount of data processed, and the resource utilization that characterizes certain operations. Considering experience, AI is able to analyze patterns and detect their changes, so it is empowered to predict future load requirements. It also can automatically adapt to its operation by changing indexes, query selectivity, and memory allocation mechanisms over time, ensuring that the database can work efficiently if workload characteristics change. Also, the AI system can identify the bottlenecks and other issues at runtime, including low-performance queries and overworked resources, and make changes that are possibly manual to be made by a human being. Indeed, the main strength of using artificial intelligence in workload analysis is based on using numeric data. There are various approaches to achieving tuning, such as reinforcement learning or employing artificial neural networks to get the mechanism of the overall performance data and even be capable of conditioning improved tuning results from one workload to another. This makes the system extremely effective and adaptive in optimizing the workloads that may be challenging for entire human DBAs to manage on a manual basis. Moreover, through the use of artificial intelligence, it is possible to minimize the possibility of making a certain mistake, provide consistent output to the organizations, and increase its resource utilization while at the same time ensuring maximum availability and availability of its services to the users.

2. Literature Review

2.1. Traditional Tuning Approaches

Historically, DBMS performance tuning has been a manual process that examines execution plans and logs and changes indexes and parameters. It is also possible to use other tools like Oracle SQL Tuning Advisor and Automatic Database Diagnostic Monitor (ADDM) to facilitate the process and present proposals based on certain rules and algorithms. Although these tools assist in automating some processes to some extent, they lack the complexity of decision-making due to the fact that they are based on predefined rules. [7-10] For this reason, the efficiency of these methods largely depends on the knowledge and feelings of an experienced DBA, which is why tuning is time-consuming and can be ambiguous.

2.2. Challenges in Traditional Methods

Nonetheless, the adverse effects of traditional tuning methods pose several critical challenges. Firstly, they show great dependency on expert DBAs, which, apart from increasing expenses, always entails a risk of an individual mistake. Secondly, usually applied machine learning approaches do not scale well and cannot operate well in different and changing workloads, and frequently, such systems have to be

recalibrated if the system's behaviour significantly changes. Also, classical tools do not possess the ability to predict performance and actively work only after a decline in performance has been observed. This reactive attribute poses a challenge in the proactive use of the existing system resources to maintain optimum performance for the long term.

2.3. AI in Database Management

In the recent past, artificial intelligence has developed into a major phenomenon in the management of databases, with potential solutions to traditional tuning approaches. Such tools as OtterTune and CDBTune are intelligent assistants using Machine learning algorithms to self-analyze workloads and databases for the best configuration. These systems use past data to create models that can suggest or enact estimable tuning as soon as possible without human interactions. Most modern tuning solutions incorporating AI techniques are unlike normal tuning techniques since they learn with time and can modify themselves, performing well in various workloads and conditions. Expert supervision is not required for their learning behaviors and greatly contributes to the stability of a system’s performance in the future.

2.4. Key Contributions from Existing Research

Several works done here are from the research level, and they have significantly developed the concept of AI-based database tuning. For instance, Zhang et al. presented an application of the reinforcement learning approach for managing system resources. They showed how the agents can improve tuning strategies in terms of the interactions between them and the environment over a period of time. Chen et al. discussed using neural networks to estimate a query performance to enable a remedy initiated when deceleration happens to the user. Furthermore, Kim et al. proposed the AI models for index recommendation that deal with one of the most time-consuming and paramount tasks related to database optimization. These works exemplify the development of AI methodologies and the possibility of using various tuning methods to outperform traditional ones.

2.5. Research Gaps Identified

Nevertheless, some issues were still not addressed during the course of the study. They found that the proposed approach consists of one notable difference: that of other systems. No generally designed tuning system includes AI for Oracle databases, even though they are considerably utilized in the current enterprise. Most solutions are designed for the open-source environment or are rather specific in their focus. Additionally, the current analytic models also do not pay adequate attention to the accuracy of workload features while employing data from current or past periods more often. As a result, they become less interactive and precise, especially when responding to dynamic activities. Last but not least, many works utilize synthetic datasets instead of actual enterprise data, making one doubt the applicability and reliability of some of the proposed solutions in real-life scenarios. All these gaps have to be filled to make progress in the form of better, more widely usable, and more broadly tuning systems.

3. Methodology

3.1. System Architecture

The proposed system architecture uses AI-based methods to automate and improve the database tuning process. It comprises five components: a Data Collection Engine, Feature Extraction, Model Training, and Bottleneck Prediction and Recommendation Engine. [11-16] Each module has a crucial function in analysing raw database metrics to precisely configure the Technical Computer Graphics Drawing module.

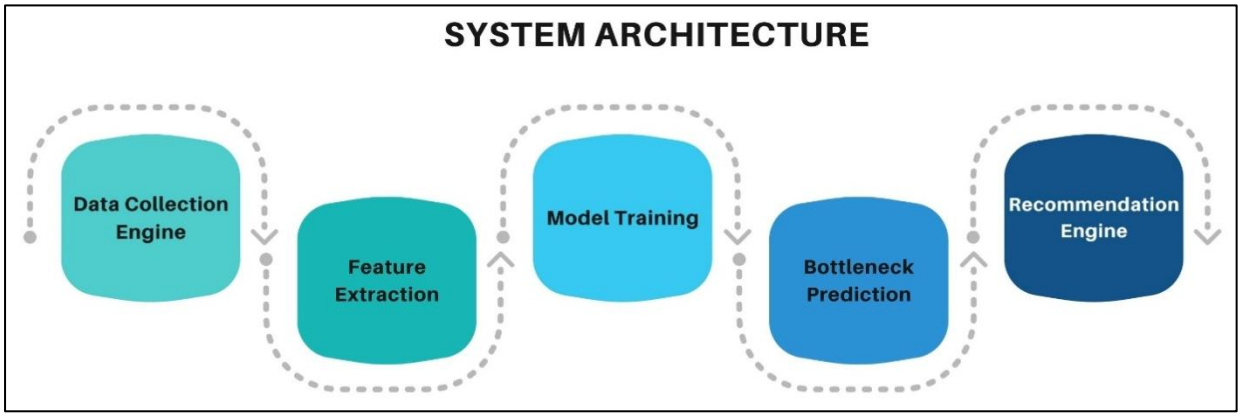

Figure 2: System Architecture

• Data Collection Engine: This is one of the primary modules of the system because it is responsible for acquiring performance-relevant data from the database environment. It periodically tracks the CPU, memory utilization, query response time, and I/O operations. It is a lightweight engine that can be set to work with performance views and monitoring tools intrinsic to Oracle or any other DB systems. This feature is very useful because it enables the subsequent stages to work with comprehensive data collected at the system and workload levels in real-time.

• Feature Extraction: After the data is obtained, the Feature Extraction module processes and prepares it to feed compatible formats to the input of the machine learning models. It entails pre-processing to include data cleaning feature selection, and if necessary, the categorical data is encoded. Specifically, query patterns, buffer cache hit ratio, and wait events will be extracted as input features. The main purpose of this module is to identify the strongest signals that affect the system's behaviour to improve the predictions made and recommendations that would be given after that.

• Model Training: The Model Training module uses records to develop a set of mathematical models that are meant to be capable of identifying the variation patterns in the optimization of the databases. The best-fit methods are used here because the methods are usually adopted from supervised learning in that explicit target values, which indicate the effectiveness of the various configurations in use, are provided. Depending on the application of the system, the learning module used may be a decision tree, a neural network, or reinforcement learning. Retraining the models periodically is very

important as they must continue to be relevant depending on the workloads in the organization.

• Bottleneck Prediction: This module effectively uses the trained models to discover current or potential hotspots in the database system. It means that monitoring real-time activities compared to the patterns it has learned minimizes cases where application contention, inefficient queries, or dub polite configurations are expected to slow down. This prediction capability enables the system to act at an instance before the performance drops significantly to support proactive and autonomous databases.

• Recommendation Engine: The last module of the architectural structure is the Recommendation Engine, which is designed to provide appropriate recommendations for fixing the discovered bottlenecks. According to the predictions and learned relationship from the training phase, it presents recommendations for resources like memory allocation, index creation, and refashioning of the query optimization plan. This way, the recommendations can be made with confidence and justification and can help DBAs decide or implement the change in a fully automated system. This one also forms the last feedback loop so that the system is constantly enhanced with minimal human interference.

3.2. Data Collection

AWR and ASH combined with the 30000 plus V$ dynamic performance views are used in the proposed system to acquire Oracle's all-around performance data. They offer detailed insight into the dynamic behavior of the database and thus allow its adequate simulation. During the data acquisition process, the following indicators are used:

Figure 3: Data Collection

• CPU Usage: This information is useful for tracking the performance of the tasks performed by the database processes related to the CPU. The high CPU load may signify that some ongoing processes have large loads that take a long time, that the database queries have not been optimized well, or that your overall hardware here is inadequate. The information on the CPU time taken by the background and user processes is found in the AWR and V$OSSTAT view, respectively. They assist in determining whether the system becomes CPU-bound and enable specific tuning strategies to improve CPU-efficient execution plans.

• I/O Latency: Input/output is described as the time needed to complete operations such as reading from and writing to the disk, and it significantly influences query speed times. There are various causes, like disk performance issues, buffer cache, or high competition in storage subsystems. With the

help of the V$FILESTAT and AWR I/O statistics, the system tracks the rates and irregularities in I/O performance. This makes disk-speed testing possible early and enables the right decision on the placement of data and configurations of storage disks.

• SQL Execution Statistics: Based on execution statistics for SQL queries, the details help when it comes to matters related to the efficiency of a particular query. This means that the above description encompasses degrees of execution time, buffer gets, rows returned, and parse counts. Derived from V$SQL and AWR SQL reports, these figures help the system to capture out-of-tune or hot SQLs. It is very useful for fine-tuning strategies like query rewriting, optimization of the database plan, or indexing to improve the performance of the database.

• Wait for Events: They define how much time individual sessions spend waiting for resources, I/O, locks, latches, or networks. It immediately indicates a conflict or resources' unavailability within the system. For instance, wait events are monitored using the information provided on the ASH reports and looking at V$SESSION_WAIT views. Such details are especially important for prioritizing tuning activities, whether at query level tuning particular query – or server level tuning server parameters.

3.3. Feature Engineering

This step is particularly important as the features will affect the basic capabilities of the developed ML model for tuning needs. In other words, in database performance tuning, features extracted from low-level performance metrics are collected from Oracle utilities like AWR, ASH, and dynamic performance views. These features cover the query behavior and the resources that are utilized by that query. A major factor is the SQL ID, which indicates a sequence number for every SQL statement processed in the database. This makes it possible for the system to monitor the results provided for a specific word to discern its usage history. With SQL ID, the buffer gets the number of times logical reads occur, demonstrating a query's usage rate of memory. The queries with a high number of buffers may be retrieving or trying to retrieve a large amount of data or have an inefficient access path, and thus, they need optimization. CPU time is the measure of how much processor time a query takes to be executed and, hence, is an essential parameter. Heavy CPU utilisation implies that the computer works hard to do calculations or that SQL statements are not optimized. Therefore, it is the same as disk reads, which give the picture of the physical I/O load the query makes to balance the storage load or utilization of the cache. High disk read ratios may indicate missing indexes or unsuitable execution plans that avoid using cache techniques. So, parse calls show how often a SQL statement is parsed and reveal cases of frequent hard parsing and insufficient use of bind variables. Parse calls should be avoided since they are resource hogs and slow down the application's performance. Also, execution count refers to the number of times a certain query is executed in the system. Among those queries that are frequently executed and yet have low response times, it is most effective to pay the most attention. Such features offer a male-to-female ratio of query activity and system effect that allows for performance forecasting

and optimal tuning instructions.

3.4. Model Development

In the model development phase, three different ML techniques are used to deal with the various parts of tuning the database performance. Every algorithm is chosen with respect to the particular problem it will solve in the system. Therefore, the system ensures a general and correct performance forecast and enhancement approach. The system employs Random Forest, K-Means, and Long Short-Term Memory (LSTM) models, each playing a distinct yet complementary role. Random Forest, which is applied for classification, is a powerful ensemble learning algorithm. [17-20] It assists in determining the chances that any particular SQL query or system configuration might cause performance problems. Through centrality on the data, which has the history of the metrics and the outcomes, the model can distinguish between the optimal queries and those prone to bottleneck queries. Random Forest has the skills to work on high-dimensional data, prevent overfitting, and give feature importance, which could be useful in determining the most informative indices and improving the model's performance. As for the case of categorizing the workloads and defining the behavioral patterns, K-means clustering is used. This family of unsupervised learning algorithms partitions sets of queries or elaborates the concept of the session in terms of resource usage characterization and organizes it with respect to parameters such as CPU, memory, and I/O. Thus, identifying these workload clusters will help the system focus more on specific tuning and increase the accuracy of model advice. It also helps find abnormally deviant queries or workloads to focus on them in more detail. To capture the temporal characteristics and to predict future states of the performance, the system employs LSTM networks, a type of recurrent neural network that is optimal for forecasting the time series. As LSTM models can capture long-term dependencies of the sequential data, they can prove useful in predicting the vulnerability to performance degradation before it happens. Thanks to the time sequence analysis, LSTM allows tuning a system ahead of time when it might become a bottleneck or expose signs of stress. The described models constitute a layered system of real-time and context-sensitive adaptive database tuning.

3.5. Recommendation Engine

The Recommendation Engine is the last and the most important part of the suggested system that aims to turn the results of the model into tuning recommendations. As such, this module provides concrete suggestions for improving database performance based on the workload analysis and performance prediction done in the previous step. The tuning options affect the upper layers of the database stack and the lower levels, and they apply to the query and system levels.

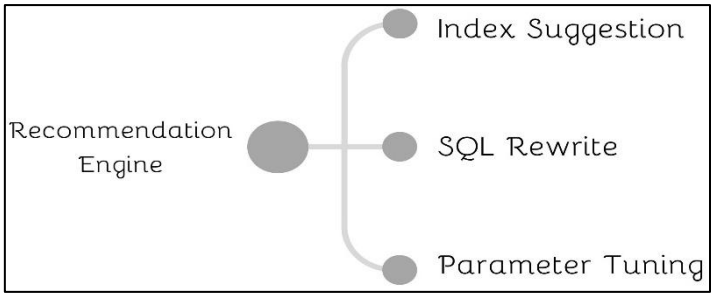

Figure 4: Recommendation Engine

• Index Suggestion: Indexing is one of the easiest ways to improve the output time, especially when reading mostly queries. Generally, it is based on detecting get-window-query and get-window-filter, join condition, and select list to determine the possibilities of reducing full table scan and I/O cost by using indexes. Based on these observations in the execution statistics and the feature importance from the build of the machine learning models, there is an indication that it is necessary to create or change B-tree or Bitmap indexes to fit the kind of workload. It may also suggest that deleting indexes is no longer necessary to increase storage and prevent specified, current indexes from disturbing DML operations.

• SQL Rewrite: One of the effective tuning techniques is to rewrite inefficient SQL statements. This way, the system is capable of recommending better queries that were observed to use high buffer gets, CPU time or disk reads constantly. The rewrites could be using bind variables, not using correlated subqueries, replacing select asterisk with a list of columns, or joining order rewriting. By changing the query structure, the engine achieves speed increases and stability of search queries on different volumes of information and in various system states.

• Parameter Tuning: At the system level, there are certain factors which need to be tuned, like the system global area, the program global area, etc. Based on historical trends and other models, the recommendation engine tries to detect preferable variations in these configurations, which should provide sufficient memory for caching, sorting and similar operations. Most of these parameters are recommended to decrease disk I/O time, reduce contention issues and enhance the workload performance, especially at periods of high activity.

3.6. Evaluation Metrics

A suitable evaluation model is used to measure the effectiveness and efficiency of the proposed AI database tuning system. Depending on the objectives of the system and the benefits accruable from the solution to the optimization problem, prediction efficiency and system improvement KPIs are utilized. These three general goals are the best means to measure the efficiency of bottleneck prediction In terms of query time. Collectively, these metrics assess to what extent the system diagnoses performance problems, fine-tunes the query usage and generally manages the use of resources. The accuracy of the bottleneck prediction is essential to assess the reliability of the developed machine learning models.

This is done following precision, recall, f1 score, and confusion matrix measures. A high value of the proposed metric denotes that the system can correctly point at queries and configurations, which would most probably lead to decreased performance. This capability is crucial for anticipatory tuning so that problems can be solved before those bugs affect the consumers or important applications. Another important key figure is the time of the query execution, which leads to improving the user time and enhancing the overall system performance. This is accomplished by calculating the mean execution time of the queries before the recommended tuning actions are implemented and the mean execution time of the same queries after the tuning action plans have been put into practice. When there is a reduction in execution times, it indicates that the recommendations, which may involve index creation, SQL rewriting or parameter tuning for the workload, improve the execution of the job.

Last, the optimisation system evaluates the CPU, memory and I/O consumption pattern to determine the effectiveness of resources after the enhancement. These metrics aggregated as CPU usage, buffer cache hit ratio, and disk I/O latency inform the system of the efficiency improvement gained with specific tuning actions. This puts it in a better position concerning scalability – and, as a result, cost as it can support more queries using less resources than before. Altogether, these values confirm the applicability and efficiency of the proposed tuning framework.

4. Results and Discussion

4.1. Experimental Setup

In the real-world environment, it is possible to realistically simulate the conditions and the characteristics of a large-scale, high-demand DBS. In this environment, the proposed AI tuning systems were evaluated. It used the Oracle 19c database system deployed on a Linux host machine with multiple cores to achieve enough processing power like the standard for enterprise-class computer systems. This was complemented with high throughput solid-state storage to achieve fast read and write, a feature fitting for modern data centers. Records briefing 1TB transactional database mimics the complex and highly sophisticated systems normally implemented in enterprises such as ERP, CRM, and financial systems. This workload description is a mixed one because it may involve OLTP (Online Transaction Processing), reporting, analytics, and batch processing, which are generic for critical business applications. The proposed workload model for the database is a genuine 30-day sample given to ensure that all the activities would be captured in a typical operation period. This dataset contained nearly all SQL statements, simple queries to complicated join statements, applying a number of aggregations, session details, duration and user transactions. Moreover, CPU usage, memory usage, disk and network traffic were also recorded during the experiment to track the effects of various system workloads. Thus, the system's behaviour was tested under various load conditions, such as high load, concurrent connections, and periods that are not very active. This way, the evaluation covered all the dynamic

aspects of the database and offered the necessary conditions for checking the efficiency of the AI-driven tuning system.

4.2. Performance Improvements

After the Tuning actions, such as the index's recommendation, SQL rewrites and parameters tuning by peculiarities of AI, the database improved on different metrics. These results suggest that using AI helped determine some of the bottlenecks and the proper setting to use, with a view to improving the system throughput.

Table 1: Performance Metrics Before and After AI Tuning

Metric	Improvement (%)
Query Response Time	58.3%
CPU Utilization	29.4%
Disk I/O Wait	66.6%

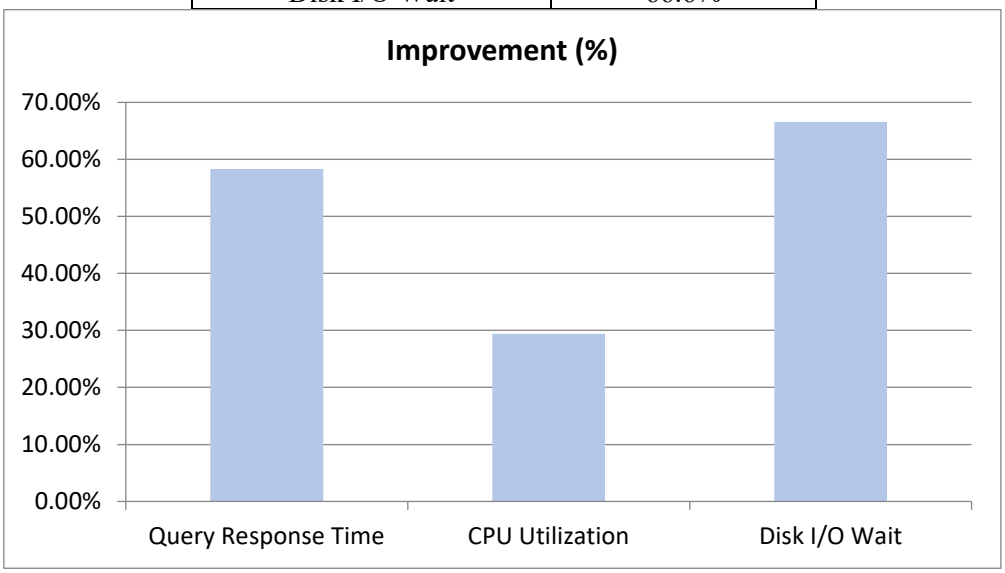

Figure 5: Graph representing Performance Metrics Before and After AI Tuning

• Query Response Time: The most promising enhancement noted was in query response time, with a 58.3% enhancement or improvement as measured. This was mainly due to increased operational efficiency by leveraging the recommendation of index recommendations and SQL alteration by the AI system. Due to this, the system ensured that creating or altering indexes for the most frequently utilized columns decreased full table scans. Also, the system provided query rewrites such as simplifying the query that involved removing certain unnecessary operations, the proper ordering of joins, and substituting subqueries with better-performing ones. Consequently, operations that took more than 1 second to search for data were brought down to a fraction of that time, and the customers noted improved navigation and optimal performance from the databases.

• CPU Utilization: In my case, the CPU utilization decreased to 29.4% through the tuning advice provided by AI optimisation. They managed to achieve this by improving the indexing of the tables and

queries used in the execution of other statements in SQL. Many queries consumed more CPU, as noted by the AI system, due to heavy computations or large data scans. This is because the system suggested index changes, as well as query rewrites that aided in simplifying these operations, hence making the database query easy and reducing its Furthermore, system and instance parameter modifications included reconfiguration of SGA as well as PGA, which were more efficient in the utilization of the CPU resources to enhance the reallocation of resource usage.

• Disk I/O Wait: However, there is a significant increase in memory usage with a very low Disk I/O wait time, which was improved by 66.6%. This reduction is an affirming sign for any AI system since it assures a positive performance in addressing I/O constraints. The system recommended indexes that can effectively replace or optimize disk reads known to have high disk I/O latency in previous queries. With the enhancement of the database schema and having checked that data, rather than being pulled from the disk, was pulled more from memory, the I/O wait times have been greatly diminished. Moreover, the settings fine-tuning that we had done, like the enlargement of the buffer cache size, frequently diminished disk contention since more data could be stored in the memory for fast retrieval. This disk I/O wait time improvement affected the total database availability and eliminated much of the lag when running queries.

4.3. Model Accuracy

Regarding the evaluation of the used machine learning models in the system, the accuracy of the models in the training and testing phases was done. Both the system for bottleneck prediction and SQL classification were very necessary in order to allow the AI-driven tuning system to discriminate between different types of performance problems and subsequently apply the correct tuning. It is also necessary to go through a detailed analysis of the accuracy measures of each model used as follows.

Table 2: Model Accuracy Metrics

Model	Accuracy (%)
Random Forest (SQL Classification)	92%
LSTM (Bottleneck Prediction)	94%
K-Means (Cluster Validation)	70%

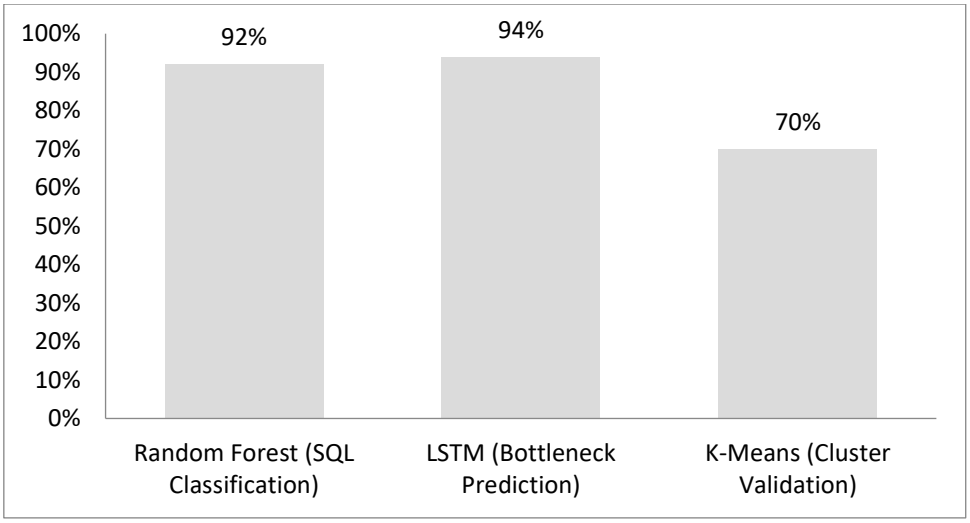

Figure 6: Graph representing Model Accuracy Metrics

• Random Forest (SQL Classification): The employment of machine learning was in using the Random Forest model whereby it was used for SQL Classification meant to determine if a given query would be optimized or otherwise, meaning that it would consume a lot of resources or take a long time to run. The results obtained while using the Random Forest model were quite promising, with the accuracy coming close to 92 per cent. This high accuracy pointed to the fact that through the analytics that goes into creating the model, the model could pinpoint between well-functioning and poorly functioning queries. Random Forest is designed to easily deal with many features and provides meaningful information regarding which query characteristics, namely CPU time, buffer gets, and I/O waits, are the most significant. This becomes beneficial for automating the classification of SQL queries into efficient and inefficient, and the proposed system recommends the appropriate tuning.

• LSTM (Bottleneck Prediction): LSTM was also used to detect bottlenecks in the system based on its data analysis of performance numbers and characteristics of workload behavior over a historical period. The LSTM model provided 94% accuracy, which shows that LSTM is effective in learner's long-term dependence on sequential datasets. This model was particularly applied well in time series; for instance, it could be used to estimate decreasing performance depending on historical data and usage rates. With such a high accuracy achieved by LSTM, the system's performance can be improved as potential bottlenecks are foreseen beforehand, and necessary precautions can be taken to prevent them. Hence, keeping track of activities such as incoming and outgoing data is an important capability to avoid system downtime.

• K-Means (Cluster Validation): As earlier mentioned, K-Means clustering does not give the classification accuracy, but the model was validated using Silhouette Score to determine the quality of the clusters since this identifies how close an object is to the cluster of its formation compared to any other cluster. Regarding the assessment of clustering, the K-Means model was used, giving a silhouette value of 0.70, which means that the clusters are well-defined. This score indicates that the workloads

are well clustered, making tuning based on the workloads easy. K Means was beneficial in pointing out certain forms of workload that may not be so evident, thus enhancing the ability to apply certain essential optimizations for various database operations.

4.4. Case Study

In order to ensure the practicality of the proposed AI-driven database tuning system, the solution was implemented in a mid-scale organization, Enterprise X, which uses an Oracle-intensive ERP system that is rather intensive in its transactions. The company's activities depended on receiving rapid, precise, and consistent solutions from databases, carrying out many daily transactions, producing reports, and storing customers' data. Originally, the database was a problem with performance issues, and certain bottlenecks in terms of workload were frequently experienced during some of those critical loads, where it took a very long time to generate reports and consumed a lot of system resources. Due to implementing the new AI tuning system, the database performance of Enterprise X was enhanced 3-fold in general. Estimations that once used to take 6 minutes to generate were completed in 2 minutes, hence enhancing operational efficiency by a vast margin. This was mainly due to the capability of the AI system to suggest alterations to indexes, transform SQL statements and adjust system parameters for efficient query execution. This proactive approach also helped identify and address bottlenecks before boundary user interaction with the IntelliQuest application. However, using this AI system also decreased the time spent on manual tuning by 70% because the tuning process was long and full of errors. A significant amount of the tuning that would have been performed manually made it possible for the DBAs to focus on other aspects, such as the architecture of the database and a deeper analysis of its performance. Another benefit was that the described system has its work presented in aspects of abilities to work without human control and thus adjust to work fluctuations. Thus, the AI system saved time for DBAs from solving such issues regularly as it tuned itself automatically in real-time and ensured maximum database efficiency at specific moments. However, this article reveals the tangible operational gains of implementing AI-driven Scala database tuning. It affirms the given system's reliability to boost performance, minimize the use of personnel's effort, and manage resources efficiently in a dynamic organization.

4.5. Discussion

These findings indicated that the proposed AI-based tuning approach provided better functionalities than conventional tuning strategies regarding flexibility, automation, and updating. On the other hand, traditional tuning utilizes ad hoc approaches that are case-based and rule-based and may need a lot of intervention, which can be time-consuming and error-prone. While the former concentrates on the bottleneck analysis of legacy performance metrics, the AI-supported approach adapts to the current performance situation accurately and addresses them by either mitigating the effects that already impacted the system or predicting and subsequently stopping new trends before they affect the quality

of the system performance. As the workload demands vary in the future, the ability to predict such changes and proactively design preventive measures is a better feature because it will guarantee maximum efficiency and effectiveness in the future than when you have not forecasted such changes. Moreover, traditional methods using set thresholds and certain assumptions about workload cannot be compared to the AI system, which develops new rules and adjusts to new workload characteristics as the data is received. The relearning process employed here also helps the system improve on the tuning tactics because the systems are also learning. This eventually results in an automatic system that only needs minimal adjustment from time to time to be in congruence with the implementation of the database. It also takes care of the present bottlenecks while increasing the efficiency of predicting and solving future performance issues. In addition, the manual adjustment is eliminated, eliminating the likelihood of making an error that is quite rampant in large data systems. There is misconfiguration, especially with dynamic workloads, because it is precipitated by human intervention. This peculiarity of the work is due to the data flow management based on a three-part model, which means that the system is classified, clustered, and capable of temporal prediction. It is compatible with different and new types of workloads, which does not affect the quality of the recommendations. This flexibility and scalability make the AI tuner an ideal solution in today's enterprise database scenarios where the need for performance tuning cannot be in a vacuum but has to adapt to anticipated and unanticipated use.

5. Conclusion and Future Work

This work shows that the proposed approach based on advanced AI may help increase the efficiency of Oracle database tuning by detecting the primary problems in areas such as memory or disk I/O operations and using machine learning algorithms to improve the tuning decisions. In our case, the system applies machine learning models for the analysis of real-time database statistics, to forecast various problems, and for tailored recommendations such as indexes, SQL query alterations, and certain system parameters. The AI-based system enhances the rate, quality and effectiveness of the database handling tasks as it does not entirely depend on manual tuning, which is normally slow, inaccurate and deterministic. In this respect, the system is capable of addressing performance issues that are already a concern at the current period and has features that will allow it to anticipate potential future threats to the performance that may emerge with the emergence of new workload patterns for the database. The efficiency of the proposed tuning system based on artificial intelligence was evaluated through a series of experiments and a real-life case, and sensible results were obtained that proved the system's ability to enhance different performance indicators, such as query response time, CPU usage, and disk input/output wait. The time reduction criteria and the capacity to minimize manual interference stresses the system's applicability to contemporary enterprises. In addition, integrating classification, clustering, and temporal prediction methods into one system guarantees the high possibility of presenting the right information and the context-awareness in recommending information to the different workloads.

Therefore, it can be concluded that the efficiency, scalability, and ability to adapt to new database workloads make AI-based database tuning an effective solution for managing the complexity of the modern DBMS.

This means that although the proposed system is highly effective, further development and enhancement can be made. It would be a good idea to focus on extending the AI tuning system to an environment with multiple tenants. Since Cloud computing and SaaS solutions are increasingly widespread, many databases share one or several tenants with different workloads and performance demands. AI could be further incorporated to personalize the tenant-specific optimizations to effectively allocate resources without causing any interference from other tenants and optimize the use of resources and performance on a large scale. One more promising direction for future research is expanding the developed approach accompanied by reinforcement learning. However, most of these machine learning models are supervised, while the case under consideration is a reinforcement system that can learn from the feedback it gets from the environment. In this way, the tuning action could change dynamically by the reinforcement learning method according to the feedback from the real-time performance tests and the change in the database workloads and environment. This would make it possible for the system to keep learning further below best strategies for optimising its trade and prevent quick stagnation that often results in poor equations of optimum means. Finally, adding an online anomaly detection module will be more advantageous. Anomaly detection would assert itself as a mechanism through which the system would be able to detect unusual attributes of performance that are either a departure from a typical workload or which could be potentially troublesome and which would lead to subsequent corrective measures in the performance before the anomalies definitively affect the performance adversely. This way, augmented real-time testing of usual database activity patterns could also help avoid glitches and thus greatly improve system performance management. Combining these features would not only improve the efficiency of the AI-based tuning system for databases, such that it would be applicable in future cases of complex and dynamic environments, but it would also make it a more powerful tool for database administrators.

References

1. Chaudhuri, S., & Narasayya, V. R. (1997, August). An efficient, cost-driven index selection tool for Microsoft SQL server. In VLDB (Vol. 97, pp. 146-155).
2. Pavlo, A., Angulo, G., Arulraj, J., Lin, H., Lin, J., Ma, L., ... & Zhang, T. (2017, January). Self-Driving Database Management Systems. In CIDR (Vol. 4, p. 1).
3. Van Aken, D., Pavlo, A., Gordon, G. J., & Zhang, B. (2017, May). Automatic database management system tuning through large-scale machine learning. In Proceedings of the 2017 ACM International Conference on Data Management (pp. 1009-1024).

4. Fekete, A., Liarokapis, D., O'Neil, E., O'Neil, P., & Shasha, D. (2005). Making snapshot isolation serializable. ACM Transactions on Database Systems (TODS), 30(2), 492-528.
5. Trummer, I., & Koch, C. (2017). Multi-objective parametric query optimization. Communications of the ACM, 60(10), 81-89.
6. Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018, May). The case for learned index structures. In Proceedings of the 2018 International Conference on Data Management (pp. 489-504).
7. Duggan, J., Elmore, A. J., Stonebraker, M., Balazinska, M., Howe, B., Kepner, J., ... & Zdonik, S. (2015). The big dawg polystore system. ACM Sigmod Record, 44(2), 11-16.
8. Li, M. L. (2023). AI-Driven Database Performance Tuning: Automated Indexing and Query Optimization. Advances in Computer Sciences, 6(1).
9. Dias, K., Ramacher, M., Shaft, U., Venkataramani, V., & Wood, G. (2005, January). Automatic Performance Diagnosis and Tuning in Oracle. In CIDR (pp. 84-94).
10. Taipalus, T., Siponen, M., & Vartiainen, T. (2018). Errors and complications in SQL query formulation. ACM Transactions on Computing Education (TOCE), 18(3), 1-29.
11. Gurry, M., & Corrigan, P. (1996). Oracle performance tuning. " O'Reilly Media, Inc.".
12. Mittra, S. S. (2006). Database performance tuning and optimization: using Oracle. Springer Science & Business Media.
13. Arb, G. I. (2024). Oracle Database Performance Improvement: Using Trustworthy Automatic Database Diagnostic Monitor Technology. Computer Science, 19(3), 881-892.
14. Nihalani, N., Silakari, S., & Motwani, M. (2009, July). Integrating artificial intelligence and database management system: an inventive approach for intelligent databases. In 2009 First International Conference on Computational Intelligence, Communication Systems and Networks (pp. 35-40). IEEE.
15. Sarker, I. H. (2022). AI-based modeling: techniques, applications and research issues towards automation, intelligent and smart systems. SN computer science, 3(2), 158.
16. Gulutzan, P., & Pelzer, T. (2003). SQL performance tuning. Addison-Wesley Professional.
17. Korotkevitch, D. (2022). SQL Server Advanced Troubleshooting and Performance Tuning: Best Practices and Techniques. " O'Reilly Media, Inc.".
18. Dageville, B., Das, D., Dias, K., Yagoub, K., Zait, M., & Ziauddin, M. (2004, August). Automatic SQL tuning in Oracle 10g. In Proceedings of the Thirtieth International Conference on Very Large Data Bases-Volume 30 (pp. 1098-1109). Gurry, M. (2002). Oracle SQL Tuning Pocket Reference: Write Efficient SQL. " O'Reilly Media, Inc.".
19. Fritchey, G. (2018). SQL Query Performance Tuning. In SQL Server 2017 Query Performance Tuning: Troubleshoot and Optimize Query Performance (pp. 1-22). Berkeley, CA: Apress.
20. Islam, S. (2024). Future Trends In SQL Databases And Big Data Analytics: Impact of Machine Learning and Artificial Intelligence. Available at SSRN 5064781.