International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V74I6P122

Volume 74 Issue 6, 309-327, June 2026

Original Article

DRAPE: A Performance-Based Database Model for Cloud Environment

Mirza Equtidar Husain[1*], Imran Hussain[1], Safdar Tanweer[1], Ihtiram Raza Khan[1]

[1]*Jamia Hamdard, Department of Computer Science & Engineering, School of Engineering Sciences & Technology, New Delhi, India.*

**Corresponding Author : mirzahusain786@gmail.com*

Received: 12 January 2026 Revised: 13 March 2026 Accepted: 18 April 2026 Published: 27 June 2026

Abstract - *Database performance is one of the most crucial topics in the IT industry. To address the performance-related issues with the Database, this research introduces DRAPE, a novel performance-based database model for the Azure cloud environment. The proposed model suggests key performance metrics configuration at the time of the database provisioning, such as resource sharing, artificial intelligence integration with DBaaS, elastic pool, and deployment model selection, database configuration like MAXDOP and QOF, along with cost optimization techniques resulting in enhancement of the database performance. Experimental results indicate a 50% reduction in query execution time, a 60% improvement in CPU compile time, an average increase of 23% in resource utilization, and an estimated cost benefit of 72%. The DRAPE is a multilevel performance evaluation system spanning database configuration, along with demonstration of the best execution plan for a query, cost savings, resource utilization in Elastic Pool, and calculation of the number of execution plans generated based on tables participating in a query using permutation theory. In contrast to existing research that mainly focuses on isolated performance key metrics, such as storage or computation. This research presents a novel paradigm that consists of a complete set of database optimization strategies with a cost-efficient design. This research is worthwhile in academia as well as in industry, demonstrating measurable performance gains across OLTP and OLAP environments. The database administrators, researchers, educators, data scientists, data analysts, and database developers stand to benefit significantly from its insights and practical applications.*

Keywords - *Cloud Computing, Cost Optimization, DBaaS, Database Performance, Elastic Pool, MAXDOP*.

1. Introduction

In the epoch of cloud computing, millions of organizations are switching to the cloud to get cost-effective and optimized services [1]. DBaaS is one of the services provided by different popular cloud vendors like Azure, AWS, and Google. The Database is one of the crucial components of any website, whether running in an OLTP or OLAP environment. Multiple techniques have been implemented and analyzed to enhance the database performance. By investigating a range of performance key metrics, including MAXDOP, Query Optimizer Fixes, resource management, index optimization, etc., improvements can be implemented and can significantly impact the efficiency and performance of the Database in cloud environments. As per the report, nowadays, data generation is increasing by leaps and bounds in the world, and in a single day, it is generating approximately 2.5 quintillion bytes. One more shocking piece of news is that 90% of the global data has already been generated within the past two years. To handle a massive amount of data, the DBMS system has been kept on changing and presented in different flavors like SQL and NoSQL. The application performance is directly connected to the underlying Database and its performance. The key characteristics that need to be given importance are cost optimization, query execution reduction time, scalability, security & flexibility, etc., in this cloud service.

The cloud providers offer services as Infrastructure/Platform/Software as a Service, termed as IaaS, PaaS, and SaaS [2, 3]. The Database, which resides in a VM (Virtual Machine), comes to an IaaS service where the user has more control on the Operating System (OS) and other administration tasks; however, if the Database is managed by the cloud provider, then there is minimal control of the user on the Database, operating system, and this service comes under PaaS. This research thoroughly investigates each database type of service provided by the cloud and presents a performance-based database model to mitigate the database performance issues.

The proposed model is DRAPE, based on SQL Server technology in the Azure SQL DB (PaaS) environment. The model contains 5 major configurations that are responsible for

the enhancement of the database performance. In the database scope, there are multiple performance metrics available to mitigate the performance-related issues; however, this paper demonstrates database-related performance metrics such as MAXDOP and Query Optimizer Fixes, along with cost savings options, Elastic Pool, and their implementation on the development environment, along with the DRAPE model architecture and workflow. Due to the paper length restriction, a few modules have been worked upon, while AI integration with the database module has been published separately. This DRAPE model stands as:

- D-Database Configuration
- R-Resource Management
- A-Artificial Intelligence
- P-Pricing Model Selection
- E-Elastic Pool Configuration

This paper introduces a novel performance-based DRAPE model, which is based on key performance metrics to boost the database performance. Secondly, the paper has presented a query execution plan generation technique, Elastic Pool cost analysis, and the implementation of two database configuration options, i.e., MAXDOP and QOF. The literature review work is given in Section 2. Section 3 provides the background and methodology. The data set and experimental setup are given in Section 4, and the results have been presented in Section 5. In conclusion, Section 6 discusses the overall outcome and future directions for working on this model.

2. Literature Review

The database performance has been a crucial topic since the advent of the Database. A million research studies have already been completed and implemented, resulting in the sharing of knowledge with the community. This research focuses on the key performance metrics responsible for enhancing the database performance. Before writing this paper, several renowned national and international journals, conference papers, blogs, and book chapters have been analyzed. The literature review summary is summarized in Table 1 below.

Table 1. Previous research work

SR. No.	Ref.	Publication Year & Source	Title of Research Paper	Key Findings/Results	Research Gap & Future Work
1	[4]	2024, Springer Nature	"Extending Your Knowledge of Azure SQL "	Theoretical concepts have been explained well, including most of the features responsible for performance enhancement.	Need to explore more and implement these features with different cloud technologies.
2	[5]	2023, IEEE	"Cloud-Based Software Development Lifecycle: A Simplified Algorithm for Cloud Service Provider Evaluation with Metric Analysis"	This paper focuses on multiple database services, service providers, and the purchasing models in Azure.	There was no practical implementation of MAXDP in this paper.
3	[6]	2022, Science Direct	"Machine Learning (ML)-centric resource management in cloud computing: A review and future directions"	The research presented a difference between traditional and ML- based cloud resource management.	It was a survey paper, and no practical implementation was presented.
4	[7]	2022, IJMLRCAI	" International Journal of Machine Learning Research in Cybersecurity and Artificial Intelligence "	This paper examines the data sharding process using an AI technique specifically for NoSQL databases.	The future work is to implement more advanced AI techniques in diverse databases in a cloud environment.
5	[8]	2020, IJIRMPS	"SQL Server Optimization-Best Practices for Maximizing Performance "	This paper provides a theoretical glimpse of all possible optimization features responsible for database performance enhancement.	Need to work on the practical implementation of the optimization features.

Bob Ward [4] has written a good paper on the awareness of SQL databases along with their fantastic features from the point of view of performance enhancement of the Database. This paper focuses on several awesome features, including the elastic pool and its importance. The elastic pool is a simple and cost-effective solution in the Azure cloud environment. The concept behind it is full utilization of all resources and saving costs for the enterprises. There is a single server hosted

in the Azure cloud, which contains multiple databases, allocating the resources as per the demand of the Database, and releasing the resources once the job finishes. The author has explained how we can get the cost savings as well as boost the performance of the Database by adding multiple databases into an elastic pool and sharing the resources. The paper published in IEEE focuses on several cloud services and service providers to enhance the database performance, along with purchasing models. Here, [5] it has been discussed how purchasing models directly impacts the database speed. Currently, there are two types of purchasing models (DTU & vCore) available in the Azure environment, which can be set up at the time of creation of any SQL DB. Each purchasing model has its own benefits and limitations and can be used as per the user's requirement. Table 2 below describes the use and limitations of each model.

Table 2. Pricing model description

Purchasing Model	Description	Applicable For
DTU-based	The Database Transaction Unit (DTU) is a bundle of compute, storage, and I/O resources involved. The computed values can be expressed either in DTUs or eDTUs.	This is a basic model and useful for those users who need simple, preconfigured resources.
vCore-based	vCore purchasing model has more flexibility in terms of choosing the compute and storage resources without any hassle. The cost saving is also possible if you use its hybrid mode.	Here, we have more control over compute and storage resources and can set them as per our requirement, which results in cost optimization.

Hemanth Gadde [6] has explored the advent of AI applications in the data sharding process in a cloud environment for NoSQL databases. He has also compared the challenges and differences between traditional data sharding and the automatic AI-enabled sharding process. This research evaluated the results received in terms of improvement of resources, such as a reduction in query latency by 15%, throughput from 50,000 to 60,000, and CPU optimization by 10%.

He has evaluated the maximum throughput by using a formula:

Throughputmax=max (i=1 ∑nLi+Wil)

Where

N = Number of shards

Li = Latency on shard i

Wi = Write load on shard i

The objective is to maintain the load across all shards, and a single shard should not be overloaded.

The author has provided a wide scope of learning in terms of resource management in traditional ways and in ML-based cloud techniques. [7] He has explained several ML-based algorithms to be used to prepare the data for pattern recognition. The paper includes all possible limitations, suggestions, and future directions related to resource management in cloud computing. The author has presented a detailed survey of the usage of the resources by Google and Azure. The underutilized usage has been published by these giants to enhance the management of the resources, a well-balanced way to reduce cost, energy, and efforts. According to the Google report, the CPU and Memory usage couldn't exceed 50% or 60%, and the rest of the usage of resources was wasted, which happened due to an imbalance in resource management. The paper also reveals the use of energy for the data centres. According to the Energy Information Administration (EIA), the consumption of energy was 35TWh in 2015, and it is expected to rise to 95TWh by 2040.

In this paper, the author tried to explain the best practices responsible for maximizing database performance in the SQL Server environment [8]. The paper sheds light on the feature configuration elaborated by the author, including DO and Don't Do suggestions to streamline the baseline of the performance.

This paper provides a theoretical knowledge of optimization of the Database, which includes "MAXDOP", Index, Resource alignment (CPU & Memory), Memory configuration, Temp DB configuration, etc. On the other hand, the paper also directs us not to use the options that degrade the performance, and these options are like not using Select *, not using many join operators in a single query, avoiding cursors, and aggregate functions like SUM (), Average () can be more expensive if not handled properly.

3. Background

The metrics, which are a part of this DRAPE model, have been classified into two groups.

- Server-level Performance Metric
- Database-level Performance Metric

Server Level Performance Metrics: These metrics need to be defined while creating a database instance in the cloud environment and will be applied overall on the instance and database level. These metrics include pricing level determination, resource configuration, deployment model as per the requirement, and automatic tuning enablement.

3.1. Pricing Model (DTU & vCore)

There are two purchasing models for Azure SQL Database: vCore and DTU [9, 10]. These models can be chosen as per customer needs and application functionality. Both models have been summarized in the following paragraph.

DTU Model: DTU stands for database transaction unit. This model is simple, with fewer good options along with a fixed price. It is suitable for those customers who need straightforward pricing. This model offers a mixed measure of CPU, Memory, reads, and writes.

vCore Model: This is another purchasing model for DBaaS services in Azure, which allows us to choose compute and storage resources independently. Cost can be saved by choosing this model along with its hybrid benefit. It is suitable for those customers who want more control over the resources.

Azure provides different service tiers for both DTU and vCore-based purchasing models, and performance depends on the chosen service tiers when creating the database instance. The service tiers for the DTU purchasing model have been elaborated in Section 3.3 and in Table 3.

3.2. Resource Selection (CPU, Memory & Storage)

As defined in Section 3.1, [11, 12], resource configuration depends on the purchasing model and service tiers assigned to it. However, more control over resources is in the vCore purchasing model compared to DTU. Here, the description of resource management is depicted in this section of the model, specifically for the Azure PaaS database service. This section also describes the situations when the defined limits of the resources are reached, and how governance mechanisms are used to apply these limits.

CPU: This resource is one of the most prominent resources in the performance enhancement of the Database. Whenever CPU usage becomes very high, query latency increases, and this can even lead to a timeout. To deal with such situations, we need to make the right decision while allocating it to the database systems.

Storage: Storage is a crucial resource that holds the data ingested into the databases. Allocation of this resource is really important, and it should be taken care of smartly. When the data space reaches its max limit, then two DML operations get impacted, and these are INSERT and UPDATE, while there is no effect on SELECT and DELETE.

Memory: As per memory architecture, memory is one of the crucial components responsible for enhancing query execution. The SQL Server relational database engine consumes most of the available memory on the database instance. The concept behind the consumption of memory is that it caches the data and provides faster read compared to storage read, which is a slow process.

3.3. Database Deployment Model (Elastic Pool)

Microsoft has introduced a basic & cost-effective technique to manage multiple databases as per the usage requirement. The databases participating in an elastic pool reside on a single server and share the resources at a set price [13]. This solution is useful and optimizes the cost, along with providing maximum resource usage. The sharing of the resources in the pool enhances the performance of the individual Database. The solution works on the demand for the resources they need when they need them and releases the resources when the requirement is over. There are ample benefits associated with this technique, such as cost savings, maximum resource utilization, etc. The implementation has been performed on the development environment for such a metric, and we saved $444 annually by swapping the service tiers in an Elastic Pool. The results of this implementation have been given in the results section of the same paper. The service tier information for the purchasing model Database Transaction Unit (DTU) is mentioned in Table 3.

Table 3. DTU service tier

Basic	Standard	Premium
Used for Development and Production	Used for Development and Production	Used for Development and Production
CPU: Low	CPU: Low, Medium, High	CPU: Medium, High
IOPS: 1-4 IOPS per DTU	IOPS: 1-4 IOPS per DTU	>25 IOPS per DTU
IO Latency: 5 ms (read), 10 ms (write)	IO Latency: 5 ms (read), 10 ms (write)	2 ms (read/write)

3.4. Artificial Intelligence Configuration

The introduction of Artificial Intelligence has become a miracle in the IT industry [14]. AI analyzes the data, schemas, queries, and workload, and provides recommendations in terms of adding/dropping the indexes from the point of view of database performance enhancement. It recognizes the pattern of the data and suggests the best optimization plan.

The current IT epoch, identified as the cloud era, is growing by leaps and bounds [15]. Thousands of organizations have already shifted from On-Premises to a cloud environment to use the well-built cloud features. Also, the integration of artificial intelligence and cloud technology provides new possibilities that are actively working to improve current cloud technologies. The big IT giants are rapidly using AI to perform difficult tasks that a few decades ago only human beings were capable of doing. To foster Artificial Intelligence diffusion, major cloud providers like Azure, AWS, and Google have presented Machine Learning (ML), deep learning analytics, and inference as a service to the

users. The AĪ enablement with DBaaS includes several benefits, such as automating routine tasks, performance improvement, and security enhancement, facilitating decision-making, and automatic tuning. The following DRAPE model uses this feature to tune the query in terms of removing the redundant indexes automatically, resulting in benefits like space reduction and database performance improvement. The practical implementation has been set up, performed, and the results have been demonstrated in Section 5.4.

Database Level Performance Metrics: There are several metrics for the database performance improvement that exist; however, as per this model, only two metrics, MAXDOP and Query Optimizer Fixes, have been included here in Section 3.5.

3.5. Database Performance Options

There are numerous database-level performance metrics available to boost the database performance; however, as per the DRAPE model, MAXDOP and QOF metrics have been discussed from the point of view of database administration.
MAXDOP: [16-19] The MAXDOP is a server/database configuration option that can be set for running SQL Server on several CPUs. This feature is responsible for the allocation of processors used to execute a single SQL statement. SQL Server can use all processors available as per the defined value set, which is "0" in the case of an on-premises/Managed instance in a Cloud environment. If the MAXDOP value increases, it results in more parallel threads per query and speeds up the query execution. The default value of MAXDOP is "8" for Azure SQL database.

To check the configuration of MAXDOP, it can be set by executing the stored procedure on SSMS, "sp_configure."
Key Points about MAXDOP:

- MAXDOP stands for maximum degree of parallelismThis value should be equal to or less than the CPU cores allocated to the SQL Server instance.
- No need to restart the server after changing the MAXDOP value
- The MAXDOP feature is available with SQL Server 2019 version and can be set at the time of installing the SQL Server instance.

Recommendations to configure MAXDOP value:

There are certain rules that need to be followed while configuring the MAXDOP mentioned in Table 4.

Table 4. MAXDOP rule

Server Configuration	Processor Count	Rule
Single NUMA node	<=8 logical processors	MAXDOP should be at or under the number of logical processors
Single NUMA node	>8 logical processors	MAXDOP should be 8
Multiple NUMA nodes	<=16 logical processors	MAXDOP should be at or under the number of logical processors
Multiple NUMA nodes	>16 logical processors	In this case, the MAXDOP value can be set to half of the processor count, with a maximum value of 16.

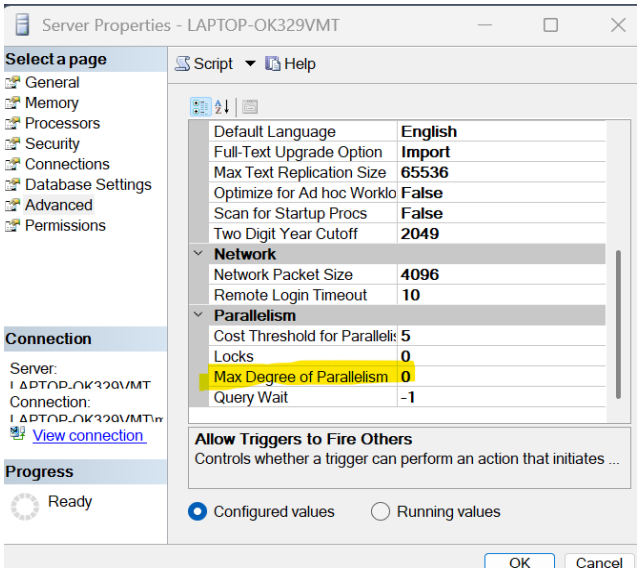

Fig. 1 MAXDOP option

In Figure 1, this MAXDOP feature is available with its default value. The snapshot has been taken from the Azure SQL VM for more visibility for the user.

3.5.1. Query Optimizer Fixes

This is another database-level performance metric that is used to improve overall system performance, including query execution time, along with minimizing resource consumption [20-22]. In the SQL Server database environment, this metric is responsible for efficiently consuming resources, aligning join operations, and reducing unwanted data retrieval.

Points to Ponder about Query Optimizer (QO):

- Query Optimizer Fixes is released by Microsoft with SQL Server 2016
- This feature produces multiple ways to execute a query and choose the plan with the optimal cost in terms of time and resources.

- This feature enhances the performance and bug fixes of the query optimizer
- Query Optimizer fixes contain the updated hotfixes, cumulative updates, or any service packs that result in performance enhancement
- There is a challenge that sometimes this feature fails to select the optimal plan due to limitations or bugs.
- It is advised to test this feature first in a non-production environment
- In a nutshell, Query Optimizer Fixes (QOF) is a treasured tool and assures the latest performance improvements and bug fixes that lead to overall database speedup.

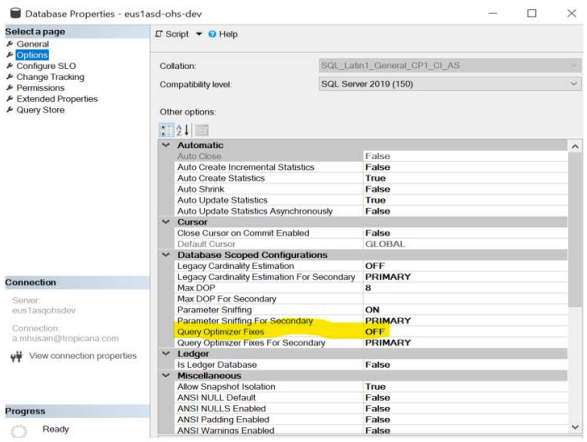

Fig. 2 QOF option

In Figure 2, the QOF feature is available with its default state. The snapshot has been taken from Azure SQLDB (PaaS) for more visibility for the user.

4. Methodology

The methodology section explores the DRAPE model working style, depicting its core key metrics along with the execution plan generation technique. This section is divided into three different sub-sections: DRAPE model architecture, Workflow diagram, and the best execution plan generation technique.

4.1. DRAPE - Framework

Cloud computing can be understood in such a way that the owner and tenant share the infrastructure. The owners are cloud operators like Azure, AWS, and Google, while tenants are enterprises, individuals, etc. The goal of cloud computing is to provide an easy and cost-effective solution; however, it faces a few challenges too. Regarding the content of this paper, the most important challenge is how to optimize database performance and provide a cost optimization solution. Therefore, we have introduced a performance-based database model to enhance the database performance in the cloud environment. We present our approach called DRAPE, which consists of five performance metrics in the cloud environment to avoid any performance issues in the live environment. Additionally, sharing infrastructure through an elastic pool and administration has clear economic benefits.

This DRAPE model focuses mainly on five prominent performance metrics, which are the catalyst to boost the database performance and avoid any similar issues in the live environment.

The performance metrics have been elaborated in brief in the background section; however, the practical implementation is listed here in the methodology and result sections, respectively. DRAPE model provides the assurance of database performance enhancement and a new direction to configure the key metrics before and after creation of the SQLDB instance to avoid any real-time performance issues in any user database. This model stands as

D: Database Configuration
R: Resource Management (CPU, Memory & Storage)
A: Artificial Intelligence Integration
P: Pricing Model Selection
E: Elastic Pool Configuration

The DRAPE model for the MSSQL Server database technology is based on key performance metrics in the Azure cloud to enhance the database performance. The outcome of this model is highly impactful and profitable to the users worldwide.

Figure 3 depicts the workflow of this model. The database administrators are responsible for any incoming user requests for database creation. Initially, the probing will be conducted through any communication method, e.g., email or chat, then the request will be analyzed from the point of view of the outcome. Here, the DBA needs to explore the ways of working to accomplish the task. As per the model's suggestion, the DBA can follow the steps as mentioned in the above diagram and configure the options. A few options are mandatory to configure while creating the instances, like "Pricing Model Selection". "Resource Selection" and "Deployment Model" are optional and can be configured once the DB instance is created.

However, the optional options are the catalyst to boost the database performance. In the result section below, the demonstration of the database configuration option, i.e., MAXDOP and Query Optimizer, and their implementation have been shown along with the results. Following this DRAPE model, the created Database will be more effective and free from any performance issues that may be encountered in the future.

4.1.1. Symbol Used

Table 5 depicts the symbol used in the DRAPE model. Each symbol has a unique meaning and represents the actual workflow in the model.

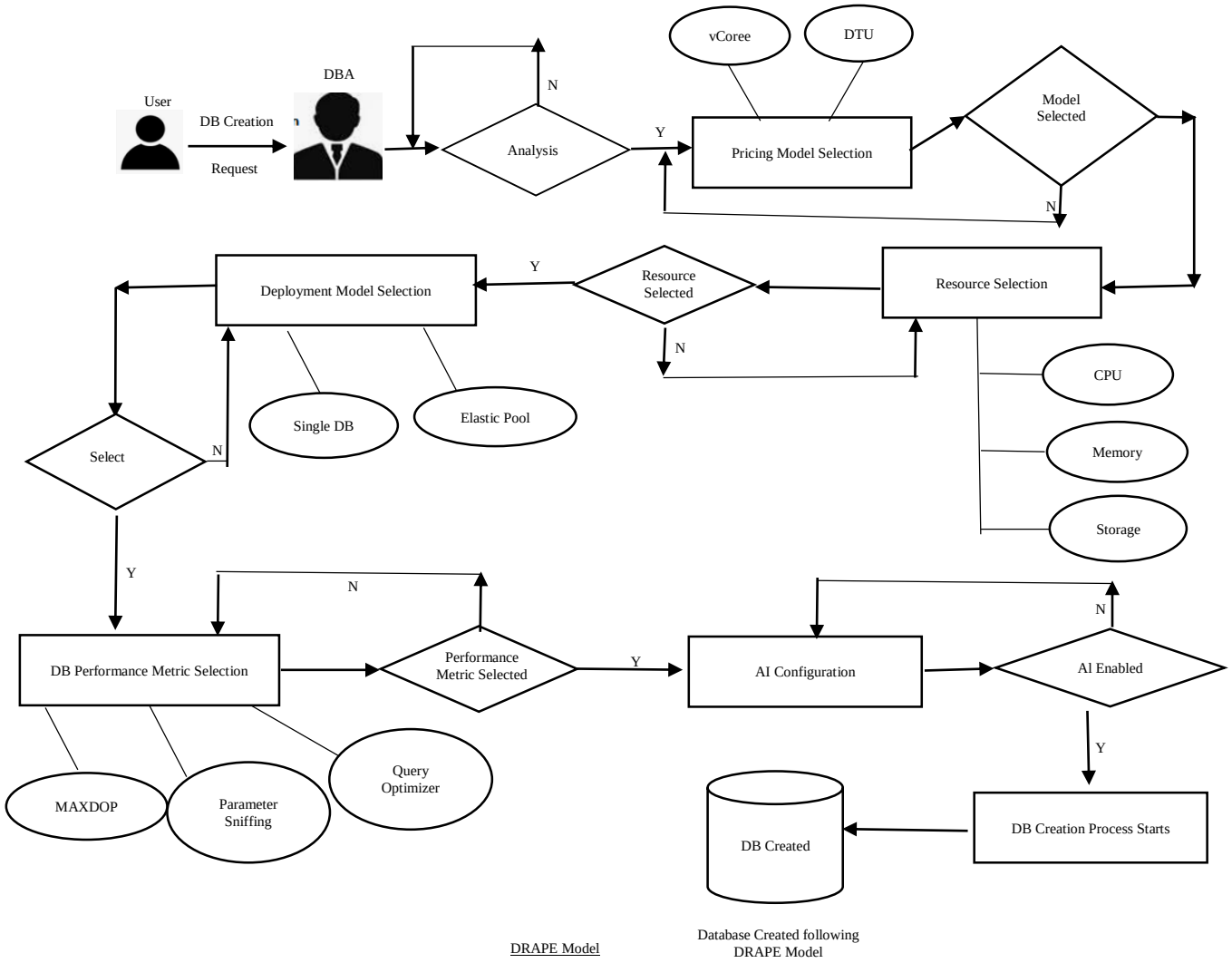

Fig. 3 DRAPE model

Table 5. Symbol used in DRAPE

Symbol	Meaning
Y	Yes
N	No
	Decision Box
	Entity Box
	Attribute Box
	Line
	Database

4.2. Data Set, Tools, Hardware, and Software Requirements for DRAPE

To set up the DRAPE model, there are hardware and software prerequisites that need to be followed. The details regarding the hardware and software are mentioned in Table 6 [Appendix]. The cloud environment is chosen as Microsoft Azure with its PaaS service, along with the latest edition and version of SQL Server and Azure underlying operating system. The SQLDB has been established in the East US location, along with the DTU purchasing model and storage of 2GB as per the Basic service tier. Moving to the database configuration details, the Microsoft sample user database "Adventure Works 2012" with a size of 228.06 MB has been chosen to work on. This Database has a total of 71 tables,

while only three tables have been used to test the query execution as per different MAXDOP values configured. The result shows that query execution time gets reduced if we set proper MAXDOP as per the workload. The results details have been mentioned in the results section along with a table and a pictorial graph.

4.3. Database Creation Workflow

To perform any task on a computer, an Algorithm is necessary, which contains the steps to perform the execution and achieve the desired result.

Here, as per the model workflow diagram, it contains inputs like (DB name, pricing model, resource values, deployment model, along with database configuration MAXDOP and QOF) as input parameters.

All these parameters will be validated at each stage. If the information is found correct, then the database creation process starts; otherwise, the process is reverted back, as seen in Figure 4.

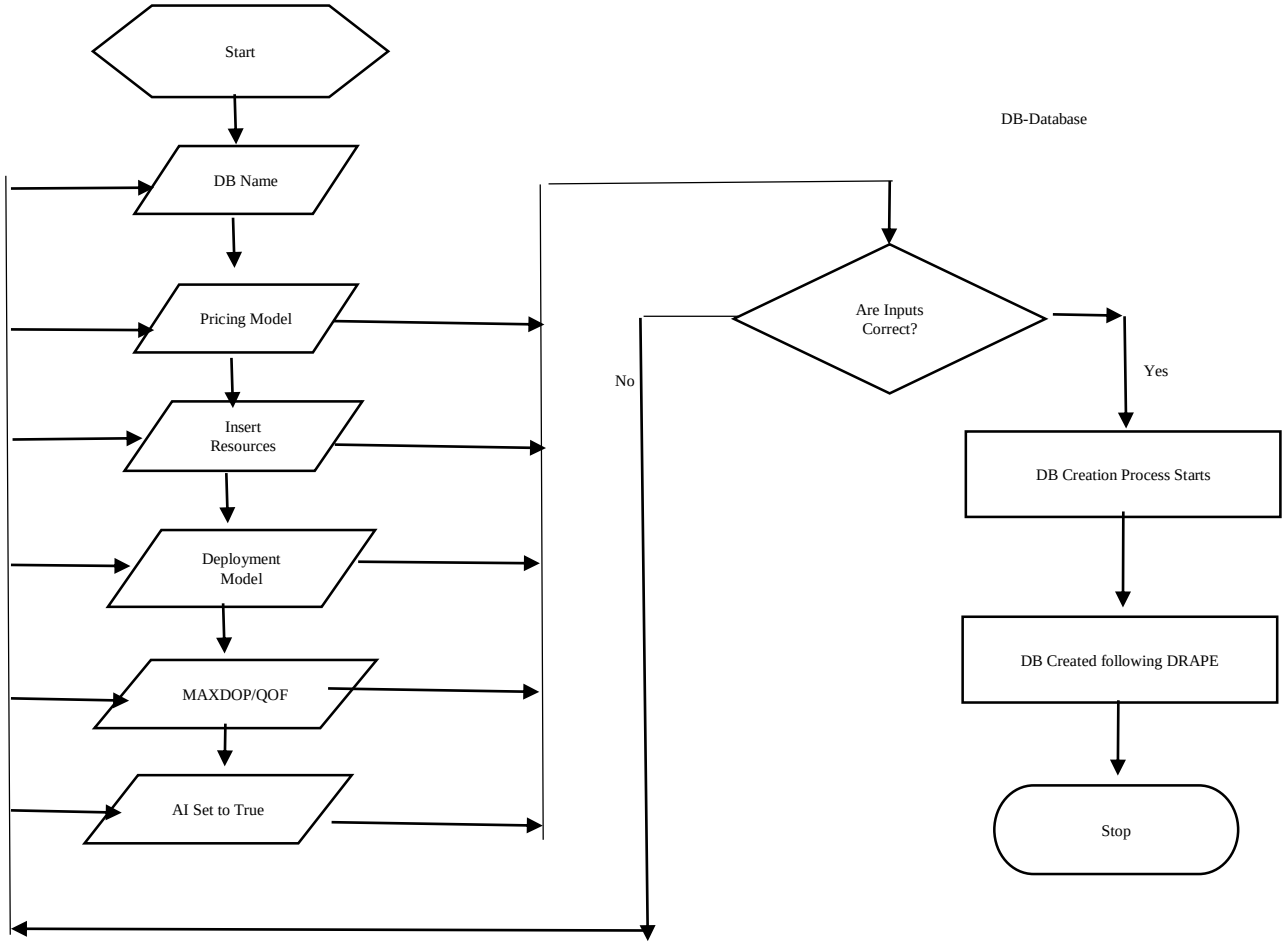

Fig. 4 Database creation workflow diagram

4.4. Methodology to Choose the Best Execution Plan

The Query Optimizer (QO) is a key factor in the SQL Server relational database management system. Whenever any query is submitted, first, it is analysed by the query optimizer along with index and table statistics availability to choose the optimal execution plan. There are several factors considered to perform this, and these factors are as follows:

- Index: Need to decide whether to use existing indexes or perform a table scan.
- Join Types: Need to choose join algorithms (Nested Loops, Hash & Merge Join) for joining tables.
- Operators Used: Operators are needed, e.g., sorts, aggregations & filters
- Cost Estimation: Each operation costs something; therefore, choosing the plan with the lowest cost

It is not a simple task to choose the best plan to execute any query; the query optimizer explores a lot of plans before choosing the best one. There are a few factors that affect this process. The factors are explained below:

- Table Join Order 2. Data Retrieval Method 3. Physical Access method 4. Physical Join Operators

4.4.1. Table Joining Order

For example, we have three tables as T1, T2 & T3, and the orders would be like this

- Join T1 with T2 and the result with T3
- Join T1 with T3 and the result with T2
- Join T2 with T3 and the result with T1

Use of Heap or Binary Tree

Use this access structure to retrieve the rows from a table

- Clustered Index
- Non-clustered Index1
- Non-clustered Index2
- Table Heap

Physical Access Method

- Index Seek
- Index Scan
- Table Scan
- Physical Join Operators
- Nested Loop (NJ)
- Hash Join (HJ)
- Merge Join (MJ)

4.4.2. Example

In this example, there are three tables (t1, t2, and t3) used, and each table has a clustered and non-clustered index. Initially, we need to check the join types and their order. Two joins are participating along with three physical join possibilities as mentioned above, and the query can be executed in 3^2=9 ways.

1. NJ –NJ
2. NJ-HJ
3. NJ-MJ
4. HJ-NJ
5. HJ-HJ
6. HJ-MJ
7. MJ-NJ
8. MJ-HJ
9. MJ-MJ

Now, check on the join order that was calculated with the help of the permutation P (n, r). Where, n = the total number of tables, r = the number of tables being arranged in a specific order. Next, we need to check the index, which is used to retrieve the data. Here we have 3 tables and two access methods: seek and scan. For each table, we have 2^2=4 choices. Here, we have 3 tables so that the choices would be 4^3=64.

Calculation 1: No of possible execution plans in case of 3 tables
No of tables (n) =3
Join Types= $3^{n-1}=3^2=9$
Join Orders = nPr = n! / (n-r)! = 3! / 1!=3
Index types with seek and scan method= $4^n=4^3=64$
Total no of possible execution plan= 3^{n-1}* nPr * 4^n=9 *3*64= 1728

Calculation 2: No of possible execution plans in case of 4 tables
No of tables (n) =4
Join Types= $3^{4-1}=3^3=27$
Join Orders = nPr = n! / (n-r)! = 4!/ 1!=24
Index types with seek and scan method= $4^n=4^4=256$
Total no of possible execution plans = 3^{n-1}* nPr * 4^n=27 *24*256= 1, 65,888

Calculation 3: No of possible execution plans in case of 5 tables
If n=5, then the calculations would look like this
Join Types: $3^{n-1}= 3^{5-1}=3^{4=}81$
Join Orders: nPr = n! / (n-r)! = 5! / 2!=60
Index types with seek and scan method= $4^n=4^5=1024$
Total no of possible execution plan= 3^{n-1}* nPr * 4^n=81 *60*1024= 49, 76,640

Here, calculations 1, 2, and 3 show the possible execution plan generations as per the number of tables participating in a query. The SQL Server engine checks all these plans and selects the best and optimal plan for the current query.

Therefore, it is important to choose the right database configuration options (index and statistics) to configure the Database, which can enhance the database performance and reduce manual as well as machine efforts.

The purpose of the demonstration of this execution plan generation is to show how much effort is carried out to execute a single statement to achieve the desired result.

The loops and access methods used in the methodology section have been demonstrated in the result section with different execution plans, along with the missing index option, which can drastically reduce query cost and execution time if implemented correctly.

5. Result & Analysis

The experimental setup has been completed in Azure Cloud with a database PaaS service. As per the model, the experiments have already been performed for other performance metrics like Artificial Intelligence configuration, Elastic Pool & resource usage, etc., in previously published papers. Due to paper length restrictions, the whole model experiment is not possible in a single paper; therefore, the implementation summary has been included in the results section for more understanding. Here, in the current setup, the experiments have been carried out on the database configuration options like MAXDOP,

QOF, the cost savings option, Elastic Pool, and usage of AI with SQLDB. So, the tables and figures in this section contain the results achieved by implementing the metrics as defined above.

The results are impactful and would be more beneficial if the same setup could be implemented in a live environment. This section is divided into four sub-sections: MAXDOP, QOF, Cost analysis in an Elastic Pool, and Space reduction & cost Savings through AI, as 5.1, 5.2, 5.3, and 5.4, respectively.

5.1. Experimental Setup: Maximum Degree of Parallelism (MAXDOP)

The practical has been performed for the MAXDOP configuration on IaaS services of Azure for SQL Server database technology as per the experimental setup mentioned in Table 7.

Table 7. Azure cloud experimental details

Database Technology	Cloud Vendor	Service	Location	Database Name	No of Processors	Default MAXDOP	Environment
SQL Server	Microsoft Azure	IaaS	East US	AdventureWorks2012	8	0	Development

Initially, we have set up an environment, and then proceed with further query execution with or without MAXDOP configuration at the database level to analyze the performance achieved. The query below has been executed in the query area of the relational database management system "SQL Server 2019". The outcome of this execution provides different results as per the values of MAXDOP, and these results are mentioned in the table below.

5.1.1. T-SQL Query

```
"SELECT
   SOH.SalesOrderID,
   SOH.OrderDate,
   C.CustomerID,
   P.Name AS ProductName,
   SOD.OrderQty,
   SOD.UnitPrice,
   SOD.LineTotal
FROM Sales.SalesOrderHeader AS SOH
JOIN Sales.SalesOrderDetail AS SOD ON SOH.SalesOrderID = SOD.SalesOrderID
JOIN Production. Product AS P ON SOD.ProductID = P.ProductID
JOIN Sales. Customer AS C ON SOH.CustomerID = C.CustomerID
WHERE SOH.OrderDate BETWEEN '2012-01-01' AND '2012-12-31'
ORDER BY SOH.OrderDate DESC, SOH.SalesOrderID
Option (MAXDOP 4)"
```

5.1.2. Execution Plan Generation

There are multiple executions, as seen in Figure 5 that have been performed on the same query with different MAXDOP values mentioned in Table 8. Here, we present the query and execution plan for a single query due to restricted paper length. These practices have been done on a single query by changing the MAXDOP values on a development environment; however, the results will be varied and more suitable on the production environment.

Table 8. MAXDOP result analysis

S. No	MAXDOP Value	No. of Statement	Query Execution Time (ms)	Statement Type	CPU Cost	Compile CPU Time (ms)	Operator Cost	Improvement %
1	0	Single	00	Select	0.133606	79	1.05229(33%)	50 % Query Reduction Time

2	2	Single	02	Select	0.123606	48	1.05229(33%)	60% Improvement in CPU compile time
3	4	Single	01	Select	0.113606	58	1.05229(33%)	
4	6	Single	00	Select	0.133606	51	1.05229(33%)	

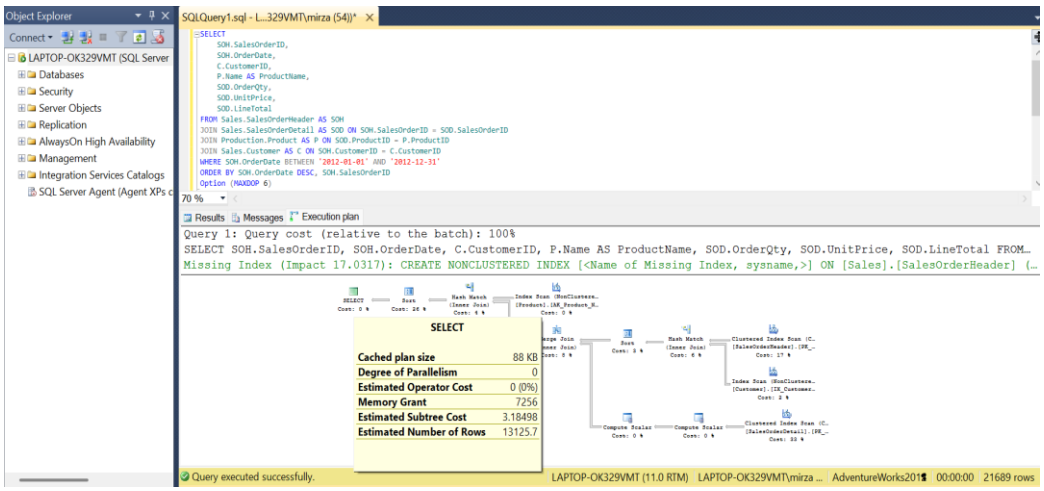

Fig. 5 Execution plan

The MAXDOP approach is more suitable in the case of multiple parallel queries running on the same server.

Here, as per the table, we can see that query execution time and CPU compile time are fluctuating along with MAXDOP values.

From the result Table 8, it is clear that if a single statement executes, then it uses all the processors available on the database server and execution is fast; however, in the case of multiple statement executions in parallel, all the queries try to use all the processors, and it will create a problem for the right alignment of the processors to the queries. Therefore, the MAXDOP feature can be used in such a scenario to limit the resource. It means the right alignment by providing a value for MAXDOP while running the query. This demonstration has used a single query and demonstrates the results by changing the value of MAXDOP.

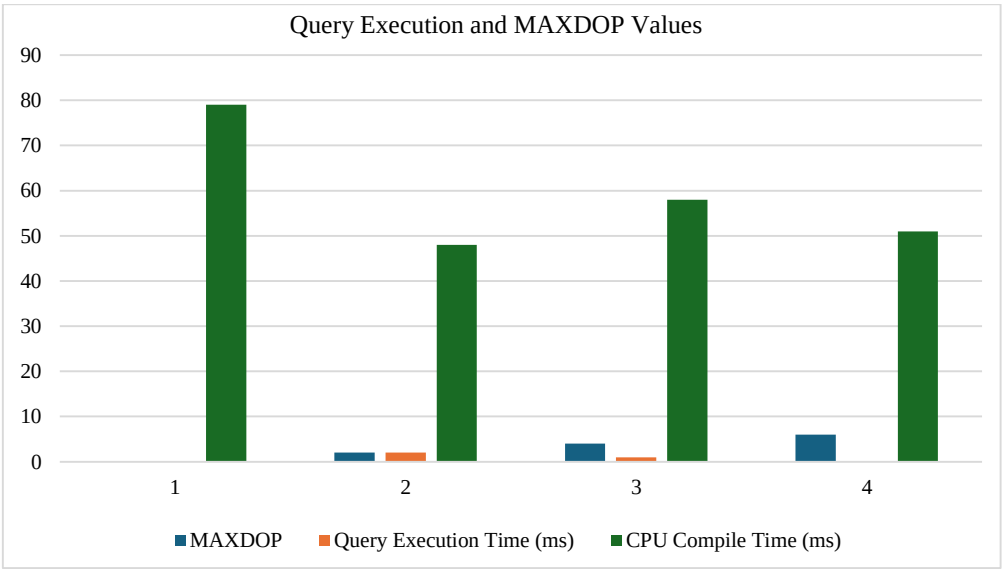

Fig. 6 Query execution and MAXDOP values

In Figure 6, the graphical representation of query execution, MAXDOP, and CPU compile time is shown. All the values related to MAXDOP, Query Execution, and CPU compile time have been taken from Table 8.

5.2. Experimental Setup: Query Optimizer Fixes (QOF)

The practical has been performed for the performance metric QOF configuration on PaaS DB services of Azure for SQL Server database technology, as per the experimental setup mentioned in Table 9.

First, the QOF will be enabled at the database level, and then the query execution will proceed further. The query has 3 tables, as mentioned in the methodology section, which contain multiple joins to fetch the records as per the condition. The query has been executed on a database. "AdventureWorks2019". As mentioned above in the methodology section, a series of multiple operations needs to be involved in the completion of the query, including operators and access methods. The execution has been performed, and the best possible execution plan has been generated as given in Figure 7.

Table 9. Experimental setup Query Optimizer Fixes (QOF)

OS Type	Database Technology	Database Name	No of Tables	Query Type	Feature
Azure OS	Azure SQL DB	AdventureWorks2019	3(Salesorderheader, Product & Customer)	Select	Query Optimizer Fixes

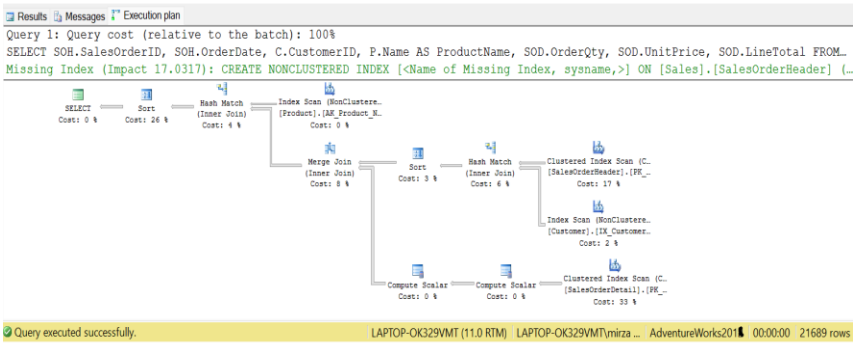

Fig. 7 Execution plan

As per the methodology description, the execution plan demonstrates the operators as Hash and Merge Joins, along with access methods like clustered index and Index scan operators. To read the execution plan, we either have to read it from top to bottom or right to left. Each operator has a cost associated with it; we need to carefully analyze it and try to reduce it as per the recommendations provided by the plan. This execution plan also gives us a recommendation for a missing index, which can further improve the execution of the query.

5.3. Experimental Setup: Elastic Pool and Cost Analysis

The practical regarding cost savings and swapping of service tier in Elastic Pool has been performed as per the configurations mentioned in Table 10.

Table 10. Azure cloud setup configuration

Cloud Vendor	Location	Service Type	Pricing Tier	OS	No of DBs	Environment
Microsoft Azure	East US	SQL DB	Basic, Standard, and Premium: 50 DTUs	Azure Managed OS	Single & 4 DBs in Elastic Pool	Development

Further in the same series of practical, facilitate the use of resource management and cost optimization, the demonstration of cost optimization has been shown by changing the service tiers. As per the DRAPE description, the deployment model will be chosen as an Elastic pool to add the Database to one pool to achieve this objective. There are a few action that needs to be taken to accomplish this target. The actions are below:

Service tier can be changed here:

- Scale DTU up or down
- Addition or removal of databases from the pool
- To find the cost summary

Figure 8 shows the Basic service tier and eDTUs assigned.

Here, in Figure 8, the basic service tier has been chosen under pool settings that contain eDTUs starting from 50 to Max 1600.

As of now, only 50 DTUs are allocated to the SQL Elastic Pool, and the estimated cost is approximately 73.62 USD per month, which is mentioned in Figure 9 for more understanding. Now, as per the requirement, the service tier can be chosen, which results in changing resources and cost. In Figure 10, the service tier is switching to Standard, and the cost estimation as per the chosen service tier is shown in Figure 11. Here, the service tier from Basic to Standard has been changed, i.e., budget-friendly, and now Figure 11 depicts changes in the eDTUs limit. The cost estimation also changes as per service tier, while only 50 DTUs are allocated.

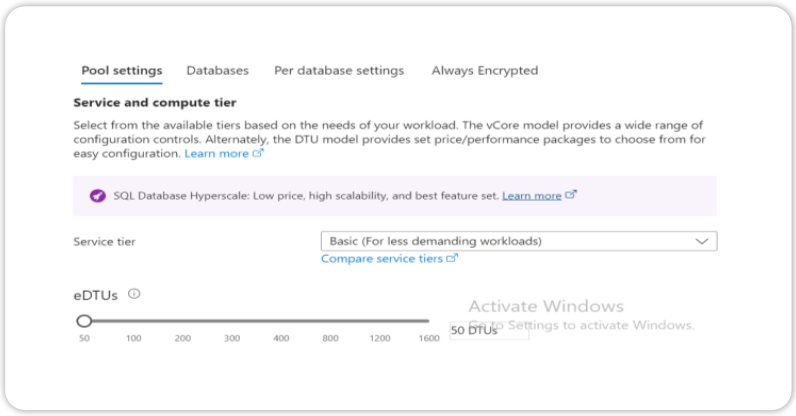

Fig. 8 Service tier & eDTUs

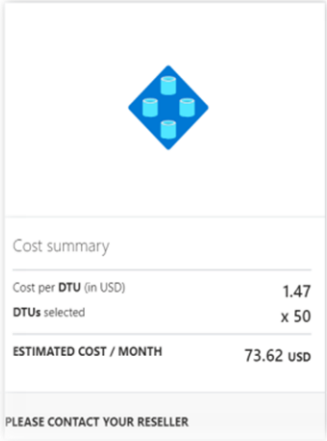

Fig. 9 Cost summary

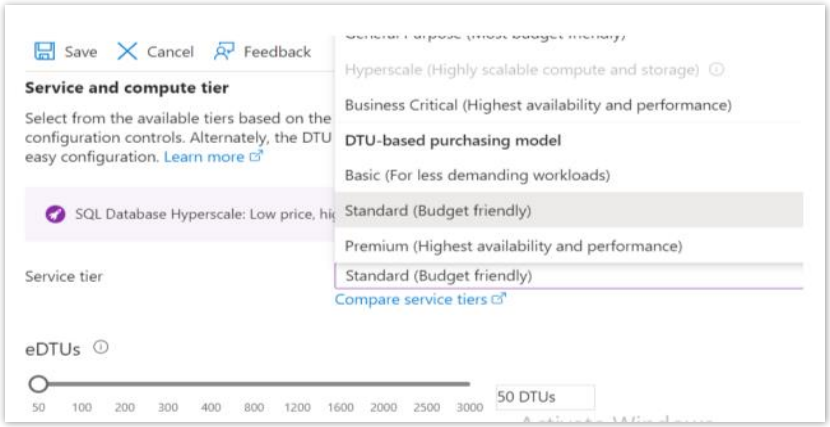

Fig. 10 Service tier switching from basic to standard

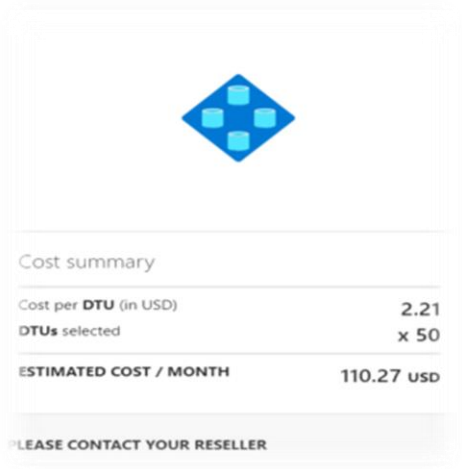

Fig. 11 Cost summary

Therefore, it is clear from the above implementation that resources can be allocated/deallocated as per the requirements, and the cost would vary. That's why it is known as an elastic pool. The elastic pool is useful from the point of view of cost savings and maximum resource utilization.

5.3.1. Cost Analysis

The following 11, 12, and 13 unfold the cost analysis and its difference by choosing appropriate service tiers as per requirements. The cost summary presented here is calculated in the Microsoft Azure environment from the SQL DB service. In both tables, Database Transaction Units (DTUs) along with other parameters are the same except for service tiers, which resulted in a benefit of approximately $37 monthly, which is calculated annually at $444. With the premium service tier, by default, 125 eDTUs are allocated initially. In Figure 12, the cost optimization chart has been presented, which shows the service tiers as Basic, Standard, and Premium, along with the cost savings in case of switching from Premium to Basic or Standard service tiers. The chart has been designed based on the results shown in Tables 11, 12, and 13.

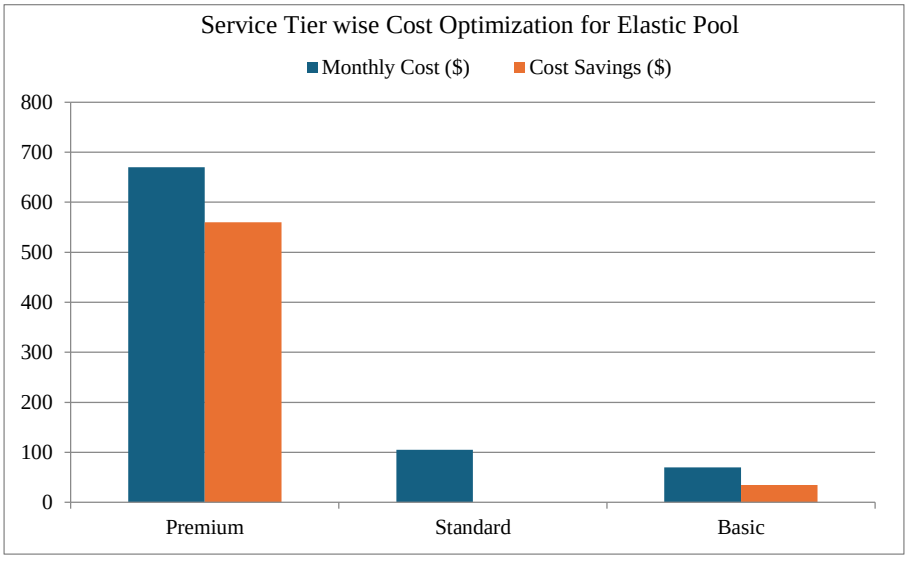

Fig. 12 Service tier wise cost optimization

Table 11. Azure cloud setup configuration with basic pricing tier

Cloud Vendor	Location	Service Type	Pricing Tier	Monthly Cost ($)
Microsoft Azure	East US	SQL DB	Basic: 50 DTUs	50 * 1.47= 73.62

Table 12. Azure cloud setup configuration with standard pricing tier

Cloud Vendor	Location	Service Type	Pricing Tier	Monthly Cost ($)
Microsoft Azure	East US	SQL DB	Standard: 50 DTUs	50* 2.21= 110.27

Table 13. Azure cloud setup configuration with premium pricing tier

Cloud Vendor	Location	Service Type	Pricing Tier	Monthly Cost ($)
Microsoft Azure	East US	SQL DB	*Premium: 125 DTUs	125* 5.41= 676.25

5.4. Experimental Setup: Space Reduction & Cost Savings through Artificial Intelligence (AI)

The scenario has been tested on Microsoft Azure, and the prerequisites of the environment are given in Table 14. The performance metric storage has been included in this setup while testing the index operation in the Azure cloud environment. By integrating AI with DBaaS SQL database, this experiment shows the space reduction and cost savings by dropping the unused indexes from the Database. Table 14 depicts the different parameters and their corresponding values, which have been used in provisioning the SQL database in the Azure environment. AI has been enabled with this. database during creation. Integration of AI provides the automatic suggestion of adding and removing the indexes to speed up the Database and reclaim the storage, too. The same scenario has been shown in Figures 13 and 14. Figure 13 depicts the recommendation for duplicate or unused indexes that need to be dropped on the Azure portal for SQL Database. Advantages of implementing the recommendation, as outlined in Figure 14, include:

Table 14. Azure cloud setup configuration (SQL database in the Azure environment)

Cloud Vendor	Location	Service Type	Pricing Tier	Storage	OS	Environment
Microsoft Azure	East US	SQL Database	Standard S1: 20 DTUs	100 GB	Azure Managed OS	Development

5.4.1. Space Optimization

Freeing up unused storage by removing redundant indices.

5.4.2. Performance Improvement

Reducing query execution times by eliminating unnecessary data overhead.

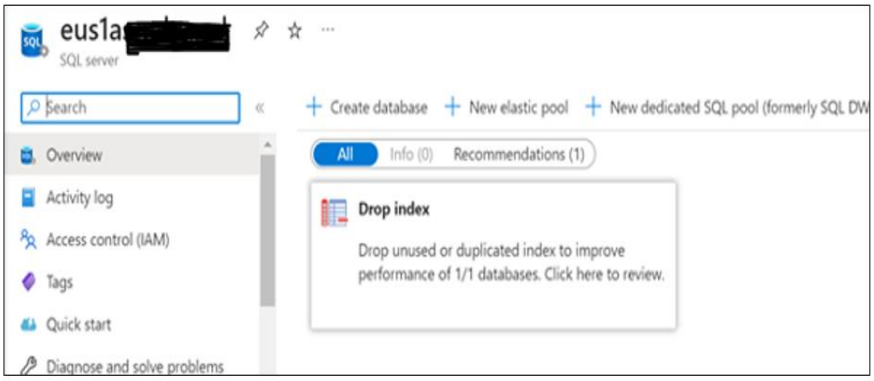

Fig. 13 Automatic tuning recommendation for drop index

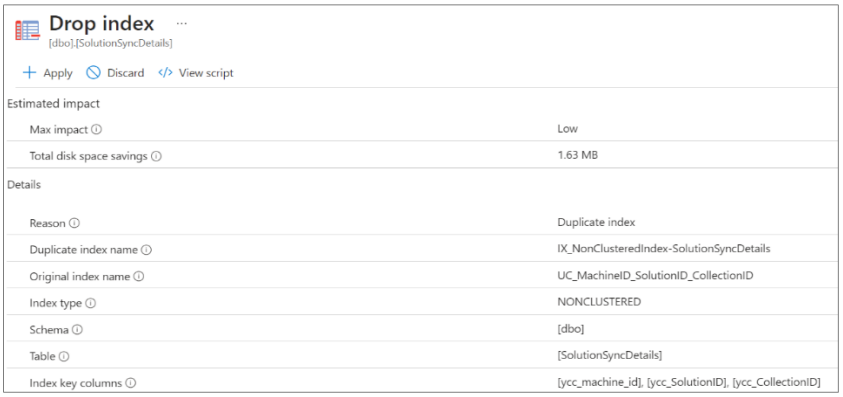

Fig. 14 Recommendation details about drop index & space saving

6. Conclusion

This research presents a novel DRAPE performance-based database model designed for a cloud environment, targeted at improving database proficiency through key performance metrics such as MAXDOP, Query Optimizer Fixes (QOF), Elastic Pool, AI integration, and resource management. The paper presents a query execution plan

generation technique and the implementation of two database configuration options, i.e., MAXDOP and QOF, along with the cost analysis for Elastic Pool in Azure IaaS and PaaS services. Furthermore, the Integration of AI with SQLDB is explored, demonstrating its role in providing automatic suggestions, including removing redundant indexes from the Database. Due to length restrictions, the full model implementation is not included here; however, module-wise work has been published separately.

As a result, there is a gain of 72% in cost savings, 23% better resource utilization, 50% improvement in query execution, 60% in CPU compile time, and an annual cost savings of $444. Initially, the implementation of this model has been performed in Microsoft Azure with SQLDB PaaS service. Future plan is to work, test, implement, and validate the same model with different cloud vendors like AWS and Google with different database technologies such as Oracle, etc. The DRAPE model delivers significant benefits to enterprises, offering AI-powered recommendations and a wider scope of performance enhancement in cloud-based database systems. It is valuable not only to DBAs but also to IT managers, researchers, educators, architects, and developers, thereby extending its impact across both academic and industrial domains.

6.1. Key Contributions

- DRAPE Model: Proposed a novel performance-based database model tailored for the Azure cloud DBaaS PaaS Service, addressing performance-related issues and their mitigation solutions.
- Query Optimization: Designed and presented a query execution plan generation technique to optimize query performance.
- Performance Metrics: Identified and discussed key database performance metrics responsible for database performance enhancement.
- Database Configuration Options: Implemented and evaluated Maximum Degree of Parallelism (MAXDOP) and Query Optimizer Fixes (QOF) to present the right choice for database configuration options.
- Resource Sharing: Demonstrated cost analysis and resource usage within Elastic Pool Configuration, leveraging the DRAPE framework for dynamic resource allocation and performance efficiency.
- AI Integration with Azure PaaS DB: Presented the usage and impact of AI techniques with PaaS database service in the Azure Environment, highlighting the automatic suggestions for removal of redundant indexes from the Database.

Competing Interests

No conflict has been declared by all the authors.

Funding Information

No grant was received for this research work from funding institutions in the public, commercial, or not-for-profit sectors.

Author Contributions

All authors contributed to the conception, design, and execution of the study. Specific contributions include: Author A: Conducted the experiments and prepared the initial draft of the manuscript. Author B: Supervised the research and reviewed the manuscript. Author C: Provided technical expertise and critical revisions to the manuscript. Author D: Reviewed the manuscript and offered valuable insights into the research findings.

Data Availability Statement

All supporting data for the study findings are contained in this article. No external databases were used; research was largely based on simulations and implementations described in the paper. However, my published paper, based on resource management, AI configuration, etc., which is a part of this model, can be available on demand.

Research Involving Human and /or Animals

Not applicable

Informed Consent

Not applicable

Abbreviation

1. DRAPE -Database Configuration, Resource Usage, Artificial Intelligence, Pricing Model, Elastic Pool
2. OLTP -Online Transaction Processing
3. OLAP- Online Analytical Processing
4. SQL-Structured Query Language
5. QO- Query Optimizer
6. QOF- Query Optimizer Fixes
7. DBaaS-Database as a Service
8. IaaS-Infrastructure as a Service
9. VM- Virtual Machine
10. OS-Operating System
11. MAXDOP-Maximum Degree of Parallelism

12. DB-Database
13. I/O-Input/Output
14. CPU-Central Processing Unit
15. MS-Millisecond
16. T-SQL-Transact-SQL Structured Query Language
17. SSMS-SQL Server Management Studio
18. TWh-Tera Watt-hour

References

[1] Janardhana Rao Sunkara et al., "Optimizing Cloud Computing Performance with Advanced DBMS Techniques: A Comparative Study," *Journal for ReAttach Therapy and Developmental Diversities*, vol. 6, no. 10s (2), pp. 2493-2502, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Martin Zbořil, and Vlasta Svatá, "Performance Comparison of Cloud Databases," *Procedia Computer Science*, vol. 256, pp. 388-395, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[3] Paolo Ferrari et al., "On the Performance of Cloud Services and Databases for Industrial IoT Scalable Applications," *Electronics*, vol. 9, no. 9, pp. 1-17, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[4] Bob Ward, *Extending your Knowledge of Azure SQL*, Azure SQL Revealed, Apress, Berkeley, CA, pp. 443-472, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[5] S. Santhosh, and Narayana Swamy Ramaiah, "Cloud-based Software Development Lifecycle: A Simplified Algorithm for Cloud Service Provider Evaluation with Metric Analysis," *Big Data Mining and Analytics*, vol. 6, no. 2, pp. 127-138, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[6] Hemanth Gadde, "AI in Dynamic Data Sharding for Optimized Performance in Large Databases," *International Journal of Machine Learning Research in Cybersecurity and Artificial Intelligence*, vol. 13, no. 1, pp. 413-440, 2022. [Google Scholar]

[7] Tahseen Khan et al., "Machine Learning (ML)-Centric Resource Management in Cloud Computing: A Review and Future Directions," *Journal of Network and Computer Applications*, vol. 204, pp. 1-51, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Krishna Kishor Tirupati et al., "Improving Database Performance with SQL Server Optimization Techniques," *Modern Dynamics: Mathematical Progressions*, vol. 1, no. 2, pp. 450-494, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[9] Serhii Minukhin, "Performance Study of the DTU Model for Relational Databases on the Azure Platform," *The Current State of Scientific Research and Technology in the Industry*, no. 1 (19), pp. 27-39, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[10] C. Gopalakrishnan et al., *Cost Management in Cloud Services for Business*, Embracing the Cloud as a Business Essential. IGI Global Scientific Publishing, pp. 63-84, 2025. [Google Scholar]

[11] Olga Poppe et al., "Proactive Resume and Pause of Resources for Microsoft Azure SQL Database Serverless," *Companion of the 2024 International Conference on Management of Data*, Association for Computing Machinery, New York, NY, United States, pp. 227-240, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[12] Yoji Yamato, "Server Selection, Configuration and Reconfiguration Technology for IaaS Cloud with Multiple Server Types," *Journal of Network and Systems Management*, vol. 26, no. 2, pp. 339-360, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[13] Manal Fadhil Younis, "Enhancing Cloud Resource Management based on Intelligent System," *Baghdad Science Journal*, vol. 21, no. 6, pp.
2156-2166, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[14] Geoff Hiten, *Plan and Implement Azure SQL Database Solutions*, Administering Microsoft Azure SQL Solutions, Apress, Berkeley, CA, pp. 21-40, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[15] Xuanhe Zhou et al., "Database Meets Artificial Intelligence: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 3, pp. 1096-1116, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[16] Punit Goel et al., "AI-Enhanced Performance Monitoring for Saas Applications in the Cloud," *2025 First International Conference on Advances in Computer Science, Electrical, Electronics, and Communication Technologies (CE2CT)*, Bhimtal, Nainital, India, pp. 1405-1410, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[17] Peter A. Carter, *Database Consistency*, Pro SQL Server 2022 Administration, Apress, Berkeley, CA, pp. 315-349, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] Vishal Vyas et al., "Managed Resource Scaling in Amazon EMR," *Companion of the 2025 International Conference on Management of Data*, Association for Computing Machinery, New York, NY, United States, pp. 662-674, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[19] Bob Ward, *SQL Server 2022 on Azure Virtual Machines*, SQL Server 2022 Revealed, Apress, Berkeley, CA, pp 413-453, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] Brian Beach, *SQL Server RDS Parameters*, Pro Powershell for Amazon Web Services, Apress, Berkeley, CA, pp 279-284, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[21] Ryan Marcus et al., "Neo: A Learned Query Optimizer," *arXiv preprint*, pp. 1-18, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[22] Srinath Shankar et al., "Query Optimization in Microsoft SQL Server PDW," *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, Association for Computing Machinery, New York, NY, United States, pp. 767-776, 2012. [CrossRef] [Google Scholar] [Publisher Link]

Appendix

Table 6. DRAPE-Pre-requisite

Hardware & Software Requirements									
Database Technology	Edition and Version	Cloud Vendor	Service	Operating System	Graphical Tool	Purchasing Model		Storage	Location
SQL Server	SQL Server Azure Edition	Azure	PaaS	Microsoft Underlying OS	SSMS	DTU	Basic Service Tier	2 GB	East US
Data Set									
Database Name	Database Size	Database Type	No. Of Tables	Table Name (Used)	No. of Records	Index (Applied)	Metric Tested	Environment	
AdventureWorks 2012	228.06 MB	Sample	71	1. Sales.Sales Order Details 2. Sales. Sales Order Header 3. Production. Product 4. Sales 5. Customer	1. Sales.SalesOrderDetails=121317. 2. Sales.salesOrderHeader=31465 3. Production.Product=504 4. Sales.Customer=19820	Clustered and Non-Clustered	MAXDOP & QOF	Development	