MIT&AIS-2026

27-29 квітня 2026
Харків-Яремче 2026

2 Міжнародна науково-практична конференція
«Сучасні інформаційні технології
та системи штучного інтелекту»
(MIT&AIS-2026)

Матеріали конференції

Частина 1

Харків 2026

Міністерство освіти та науки України
Національна академія наук України
Координаційна рада НАН України з питань штучного інтелекту
Харківський національний університет радіоелектроніки
Харківський національний університет імені В.Н. Каразіна
Карпатський національний університет імені Василя Стефаника
Північного-Східний координаційний науковий центр з питань штучного інтелекту
Інститут кібернетики імені В.М. Глушкова НАН України
Університет технологій в Лодзі
Університет Павла Йозефа Шафарика в Кошице

СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ MIT&AIS-2026

Матеріали

2-ї Міжнародної науково-практичної конференції

Частина 1

27-29 квітня 2026 р.
Харків – Яремче, Україна

Харків 2026

Development and Research of Batch Implementation of SQL-queries Based on the Rules of Their Ordering in Cloud Environments

Serhii Minukhin [a]

[a] *Simon Kuznets Kharkiv National University of Economics, Nauky Ave, 9A, Kharkiv, 61166, Ukraine*

Abstract

The study proposes strategies for grouping (batching) queries in relational databases based on random grouping, as well as on prioritizing the values of individual performance metrics – execution time, DTU usage, and CPU load – and analyses their impact on the performance of the Azure SQL Database cloud platform service. The research methodology included creating a database in Azure SQL Database at different Azure service tiers – from S0 to S12 – to model various configurations of computing resources. To simulate realistic scenarios of working with the service, a database of a trading company with large sets of test data and several test database queries of varying complexity was used. Query batching strategies were developed: random grouping, grouping by ascending/descending query execution time, resource intensity (DTU consumption), and CPU load. Each strategy was tested across all resource configurations through multiple test trials, ensuring the relevance of the obtained results for an objective analysis. The results obtained demonstrated the necessity of using a differentiated approach to selecting query batching strategies depending on database size, query complexity, and the choice of query prioritization models in batch mode.

Keywords

Azure SQL Database, Big Data, query complexity, execution time, DTU, CPU load, service tiers, ordering, batching strategies.

1. Introduction

With the development of technologies and the growth of data volumes, the efficiency of processing and managing large datasets is becoming increasingly critical. The rapid growth in the volume of data generated and processed by organizations, particularly due to the development of social networks, the Internet of Things (IoT), and multimedia, creates new challenges for database management systems. This trend, known as Big Data, is characterized not only by significant data volumes but also by their variety and velocity, which requires new approaches to their storage and processing [1]. This necessitates continuous improvement of approaches to query creation and optimization of resource usage in database management systems [2].

One of the key advantages of cloud databases, such as Azure SQL Database, is the ability to dynamically scale resources according to demand. However, to maximize the use of this flexibility, it is important to carefully plan and optimize query execution, especially when executing them massively in parallel under conditions of large data volumes. In the context of cloud computing, traditional query optimization strategies may prove insufficiently effective due to the dynamic nature of resources. This creates a need to apply adaptive approaches to query formation and processing, enabling flexible responses to changes in the availability of resources [3].

MIT&AIS-2026: 2nd International Scientific and Practical Conference "Modern Information Technologies and Artificial Intelligence Systems", April 27–29, 2026, Kharkiv-Yaremche, Ukraine
EMAIL: serhii.minukhin@hneu.net
ORCID: 0000-0002-9314-3750

2. Objective

The aim of this study is to improve the efficiency of working with relational databases by developing and implementing strategies for forming query batches within the Azure SQL Database cloud platform service and determining the most effective ones for optimizing service performance when working with big data. The study is based on the development and analysis of the impact of different query batch execution strategies on performance indicators and the efficiency of resource utilization of the cloud platform.

3. Problem Analysis

With the advancement of technologies and the growth of data volumes, there is a need to improve approaches to managing and processing large datasets. For example, the work "An Overview of Query Optimization in Relational Systems" [4] considers the main query optimization techniques, such as the use of indexes, query transformation, and the application of statistical data. These methods are aimed at improving database performance, making them particularly relevant to our study in the context of query optimization in cloud environments.

Considerable attention in the literature is given to database design issues. For instance, the work "An Approach and Software Prototype for Translation of Natural Language Business Rules into Database Structure" [5] proposes an approach and a software tool for automating the creation of database structures based on textual requirements known as business rules. Automating this process reduces the number of errors at the design stage and improves the quality of created databases, which is especially relevant given modern data volumes and system complexity.

The study "Data Dependencies for Query Optimization: A Survey" [6] analyses various approaches to optimizing SQL queries, including the use of deep data analysis, rule-based optimization, and genetic algorithms. This study demonstrates how different optimization approaches can affect query processing efficiency, which is directly relevant to our study considering the specifics of cloud databases.

The article "Optimizing Database Performance: Strategies for Efficient Query Execution and Resource Utilization" [7] examines various methods and techniques for improving database performance without compromising data integrity or security. The authors explore approaches such as query rewriting, indexing, and caching to minimize query response time and improve overall system throughput. This is particularly important for our study, which focuses on improving performance in Azure SQL Database.

The study "Query Optimization for Databases in Cloud Environment: A Survey" [8] highlights the importance of query optimization in cloud environments to reduce execution time and improve resource utilization. This underscores the relevance of our study, which aims to identify effective strategies for working with Azure SQL Database.

Recent studies, such as "Method for Optimizing SQL Queries of a Database Management System" [9], have developed new SQL query optimizations methods specifically adapted for situations where data retrieval speed deteriorates over time. This study demonstrates how new optimization approaches can accelerate data retrieval processes, which is critical for maintaining high database performance.

Despite the large number of studies, there is a gap in examining the impact of different query execution scheduling and batch processing strategies on the performance of database services in cloud environments. This study aims to fill this gap by analyzing the impact of different strategies for forming and executing query batches on performance and resource utilization in the Azure SQL Database service.

4. Methods

Setting up the Azure SQL Database environment.

For the study, a database was created in Azure SQL Database at the standard service tier (Standard) [3]. This tier allows selecting different configurations of computing resources, providing flexibility and scalability according to workload requirements.

In Azure SQL Database, resources are measured in DTUs (Database Transaction Units) [3], which represent an integrated measure of CPU, memory, and input/output operations. The higher the DTU level, the more powerful the computing resources available to the database [3]. The Standard service tier offers a wide range of DTU configurations – from the lowest level S0 (10 DTU) to the highest S12 (3,000 DTU). Each tier differs not only in the number of available DTUs but also in the amount of included and maximum storage, the maximum number of queries that can be executed simultaneously, and the maximum number of concurrent sessions [3]. For the study, five different configurations of computing resources at the Azure SQL Database Standard service tier were selected: S0, S1, S3, S7, and S12. This enables a comprehensive assessment of the impact of available DTU resources on performance when implementing different query batching strategies on scalable datasets. The lowest tier, S0 with 10 DTU, was chosen as the baseline configuration for comparison. The S1 (20 DTU) and S3 (100 DTU) tiers represented intermediate options with gradually increasing computing power. Meanwhile, the more powerful S7 (800 DTU) and S12 (3,000 DTU) tiers allow testing grouping strategies under conditions of substantial resource availability [3].

For the study, a relational database (DB) modelling the activities of a trading company [10] was used, providing a realistic environment for testing SQL query performance in various scenarios.

To optimize the research process and control costs, preliminary population of tables with generated test data totaling 1.5 GB was carried out on a local resource [11]. This approach significantly reduced data generation time and minimized the costs of using cloud computing resources. After completing data generation, migration to the Microsoft Azure platform was performed, where service performance tiers ranging from S0 to S12 were used.

Development of query batch formation strategies**.** To evaluate database performance and different approaches to query batch formation, the following strategies for query formation and execution were developed.

1. In arbitrary order (Batch 1, B1). Queries are grouped into batches arbitrarily (unsorted) without considering their characteristics. This allows evaluating system performance under conditions where queries are executed without any prior analysis or optimization and establishes a baseline for comparison with other strategies. Given the total number of combinations (8!), a particular set of queries can be selected based on a random number generator.

2. In ascending order of query execution time determined interactively (Batch 2, B2). Queries are ordered in ascending order of their processing time. Purpose: this strategy improves overall performance, as faster queries are executed earlier, which may reduce the total waiting time for the remaining tasks in the batch. This is useful in scenarios where it is important to process most batch queries as quickly as possible.

3. In descending order of execution time (Batch 3, B3). Queries are ordered in descending order of their processing time. Purpose: executing longer queries first allows assessing the impact of ‘heavy’ queries on overall service performance. This may reveal how more resource-intensive queries affect the execution of subsequent ‘lighter’ queries.

4. In ascending order of the maximum DTU value involved during query execution (Batch 4, B4). Queries are ordered by increasing resource intensity (DTU usage). Purpose: this allows identifying how different levels of DTU usage affect performance. Executing less resource-intensive queries first may reduce system load, which can be beneficial under resource-constrained conditions.

5. In descending order of the maximum DTU value (Batch 5, B5). Queries are ordered by decreasing resource intensity (DTU usage). Purpose: executing more resource-intensive queries first may clarify how heavy loads affect subsequent less resource-intensive queries. This may be effective for optimizing resource allocation within the system.

6. In ascending order of the maximum CPU usage percentage (Batch 6, B6). Queries are ordered by increasing processor load. Purpose: this helps identify how CPU utilization levels affect overall system performance. Executing less CPU-intensive queries first may reduce total batch processing time.

7. In descending order of the maximum CPU usage percentage (Batch 7, B7). Queries are ordered by decreasing processor load. Purpose: executing more CPU-intensive queries first allows assessing how heavy processor loads affect subsequent, less resource-intensive queries. This helps determine the impact of ‘heavy’ queries on overall service performance.

5. Results

Experimental validation of the proposed approach was conducted to comprehensively evaluate the effectiveness of the proposed query batch formation strategies.

To analyze the impact of the proposed query grouping strategies on database performance, the following graphs present key metrics for different Azure SQL Database configuration tiers: execution time (Figure 1), CPU usage (**Error! Reference source not found.**2), and DTU usage (Figure 3) for d ifferent batch formation strategies.

Analyzing the obtained results, the following conclusions can be drawn.

1. At lower service tiers S0 and S1, where resources are limited, the order of query execution plays a key role in achieving maximum performance. Strategies that group queries by increasing execution time or resource intensity demonstrate better results at these tiers.

2. At higher tiers S3, S7, and S12, where more resources are available, the Azure SQL Database service can efficiently distribute them among parallel queries; therefore, the order of query execution within a batch becomes a less critical factor influencing service performance.

3. Grouping queries by increasing resource intensity (DTU usage or CPU load) may provide some performance improvement at higher resource tiers. This is explained by the fact that strategic grouping of queries helps avoid sharp fluctuations in resource allocation during batch execution.

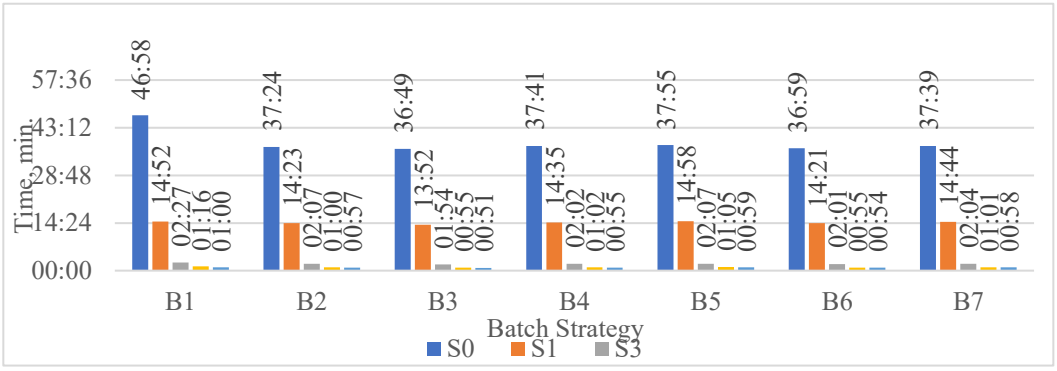

Figure 1: Batch execution time for different model tiers

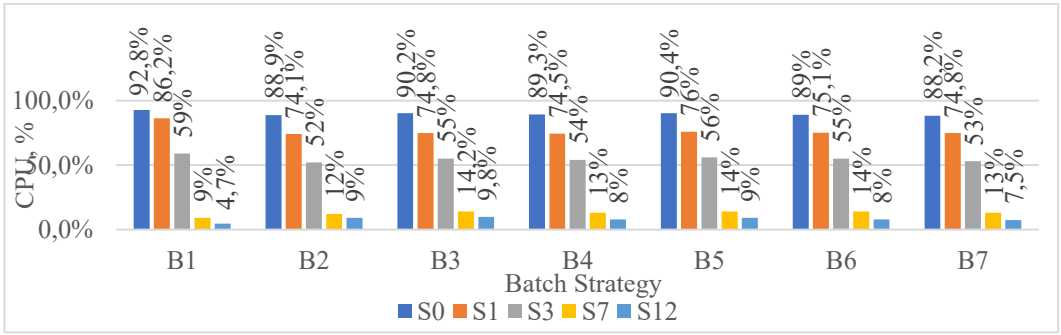

Figure 2: CPU usage for different model tiers

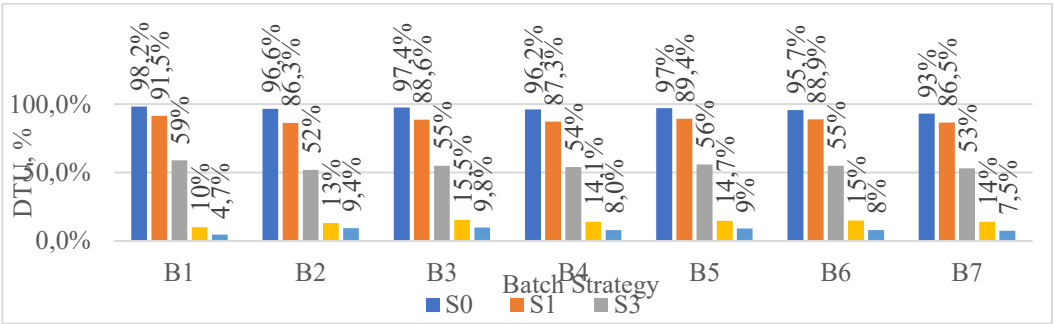

Figure 3: DTU usage for different model tiers

6. Conclusion

Within the scope of this study, the effectiveness of various query batch formation strategies in the Azure SQL Database service was developed and analyzed with the aim of optimizing its performance when working with large datasets. The experimental results confirmed that each batching strategy has a certain impact on key performance metrics – total execution time, DTU usage percentage, and CPU load percentage. Therefore, the correct choice of query batch formation strategy depends on the configuration of computing resources and the characteristics of specific queries. This may determine best practices for optimizing database performance, which can be useful for developers, database administrators, and other professionals in the field of database technologies and cloud platform services. In particular, the obtained results highlight the importance of considering the characteristics of specific tasks and the available resource provision when selecting an effective query batch formation strategy.

7. Declaration on Generative AI

The author did not employ any Generative AI tools.

8. References

[1] A. Sebaa, A. Tari, "Query optimization in cloud environments: challenges, taxonomy, and techniques," J. Supercomput., vol. 75, pp. 5420–5450, 2019. URL: doi.org/10.1007/s11227-019-02806-9.

[2] F. Ibrahim, M. Aoun, "Improving query efficiency in heterogeneous big data environments through advanced query processing techniques," J. Contemp. Healthc. Anal., vol. 6, no. 6, pp. 40–64, 2022.

[3] S. Minukhin, "Performance study of the DTU model for relational databases on the Azure platform," Innov. Technol. Sci. Solut. Ind., no. 1 (19), pp. 27–39, 2022. URL: https://doi.org/10.30837/ITSSI.2022.19.027.

[4] S. Chaudhuri, "An overview of query optimization in relational systems," in Proc. 17th ACM SIGACT-SIGMOD-SIGART Symp. Princ. Database Syst. (PODS '98), 1998, pp. 34–43. URL: https://doi.org/10.1145/275487.275492.

[5] A. Kopp, D. Orlovskyi, and S. Orekhov, "An approach and software prototype for translation of natural language business rules into database structure," COLINS, vol. 2870, pp. 1274–1291, 2021. URL: https://repository.kpi.kharkov.ua/entities/publication/67486f70-9d05-4d65-91e8-4e4c36234a14.

[6] J. Kossmann, T. Papenbrock, and F. Naumann, "Data dependencies for query optimization: a survey," VLDB J., vol. 31, no. 4, pp. 1–22, 2022. URL: https://doi.org/10.1007/s00778-021-00676-3.

[7] V. B. Ramu, "Optimizing database performance: strategies for efficient query execution and resource utilization," Int. J. Comput. Trends Technol., vol. 71, no. 7, pp. 15–21, 2023. URL: https://doi.org/10.14445/22312803/IJCTT-V71I7P103.

[8] A. Bachhav, V. Kharat, and M. Shelar, "Query optimization for databases in cloud environment: a survey," Int. J. Database Theory Appl., vol. 10, no. 6, pp. 1–12, 2017. URL: http://dx.doi.org/10.14257/ijdta.2017.10.6.01.

[9] С. В. Суліма, О. Д. Єрмолаєв, "Метод оптимізації SQL запитів системи управління базами даних," Системи управління, навігації та зв'язку, vol. 2 (72), pp. 151–157, 2023. URL: https://doi.org/10.26906/SUNZ.2023.2.151.

[10] С. Мінухін, М. Башкіров, "Моделювання роботи з базами даних торгівельних компаній на хмарних платформах," in Матеріали 12-ї Міжнародної науково-технічної конференції «Інформаційні системи та технології» (IST'23), ч. 2: Молодіжна секція, Харків, 2023, pp. 45–46.

[11] С. В. Мінухін, М. О. Башкіров, "Дослідження ефективності методів генерації тестових даних в реляційних базах даних," Системи управління, навігації та зв'язку, no. 2, pp. 127–134, 2024. URL: https://doi.org/10.26906/SUNZ.2024.2.127.