UDC 004.654:004.75 DOI https://doi.org/10.35546/kntu2078-4481.2025.4.3.28

T. A. TKACHENKO
Master's Student at the Department of Electronic Computing Machines
Kharkiv National University of Radio Electronics
ORCID: 0009-0001-1550-2586

V. O. MARTOVITSKY
Candidate of Technical Sciences, Associate Professor,
Associate Professor at the Department of Electronic Computing Machines
Kharkiv National University of Radio Electronics
ORCID: 0000-0003-2349-0578

N. M. BOLOHOVA
Associate Professor at the Department of Electronic Computing Machines
Kharkiv National University of Radio Electronics
ORCID: 0000-0001-8927-0055

METHODS OF OPTIMISING QUERIES IN DISTRIBUTED DATABASES

The article provides a comprehensive study of the problem of optimizing query performance in distributed database systems, where architectural complexity, information fragmentation, and the necessity for coordination between numerous nodes significantly affect processing speed and stability. It has been determined that conventional approaches to executing multi-table queries, especially when working with large historical data collections, do not provide acceptable response times and result in significant computational costs. In this context, it is relevant to implement specialized optimization methods that reduce the number of reading operations, the volume of intermediate calculations, and the load on the network infrastructure. The article proposes an integrated approach that combines caching, indexing, and materialization of representations to achieve the maximum performance when processing complicated analytical queries in a distributed environment. The caching provides rapid retrieval of results without accessing the disk structures, the indexing speeds up the search and filtering by creating the specialized data structures, and the materialization eliminates the need to repeat complex join and aggregation operations by storing the results in the form of physically accessible tables. For the experimental part of the research, a representative multi-table database was used, which contains information about the company's employees, their salaries, and their affiliation with departments and divisions. Five different variants of the same query were executed: without optimization, after caching, after indexing, using materialized representation, and using a combination of all three methods. The experimental results demonstrate that although each method separately provides a noticeable acceleration of performance, the greatest effect is achieved when they are used simultaneously. The query execution time was decreased from 2831.869 ms to 0.076 ms, which is an increase in performance by more than 37,000 times. The data obtained confirm the existence of a synergistic effect and demonstrate the high practical importance of comprehensive optimization for the systems that work with heavy analytical loads in distributed data processing conditions. The presented approach can be used to modernize existing information systems and increase their efficiency when scaling.

Key words: *query optimisation, distributed databases, caching, materialisation, indexing, PostgreSQL*

Т. А. ТКАЧЕНКО
магістрант кафедри електронних обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0009-0001-1550-2586

В. О. МАРТОВИЦЬКИЙ
кандидат технічних наук, доцент,
доцент кафедри електронних обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0000-0003-2349-0578

Н. М. БОЛОГОВА
доцент кафедри електронних обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0000-0001-8927-0055

МЕТОДИ ОПТИМІЗАЦІЇ ЗАПИТІВ У РОЗПОДІЛЕНИХ БАЗАХ ДАНИХ

У статті здійснюється комплексне дослідження проблеми оптимізації продуктивності виконання запитів у розподілених системах управління базами даних, де складність архітектури, фрагментація інформації та необхідність координації між численними вузлами істотно впливають на швидкодію та стійкість обробки. Визначено, що традиційні підходи до виконання багатотабличних запитів, особливо за умови роботи з великими історичними наборами даних, не забезпечують прийнятного часу відповіді та призводять до значних обчислювальних витрат. У цьому контексті актуальним є застосування спеціалізованих методів оптимізації, які зменшують кількість операцій читання, обсяг проміжних обчислень та навантаження на мережеву інфраструктуру. У роботі запропоновано інтегрований підхід, що поєднує кешування, індексацію та матеріалізацію представлень з метою досягнення максимальної швидкодії при обробці складних аналітичних запитів у розподіленому середовищі. Кешування забезпечує оперативне повторне отримання результатів без звернення до дискових структур, індексація прискорює пошук і фільтрацію за рахунок створення спеціалізованих структур даних, а матеріалізація виключає повторне виконання складних операцій об'єднання та агрегації, зберігаючи результати у формі фізично доступних таблиць. Для експериментальної частини використано реалістичну багатотабличну базу даних, що містить інформацію про працівників підприємства, їхні заробітні плати, належність до відділів і службових підрозділів. Проведено п'ять варіантів виконання одного й того самого запиту: без оптимізації, після кешування, після індексації, із застосуванням матеріалізованого представлення та за умови комбінованого використання всіх трьох методів. Результати експериментів демонструють, що хоча кожен метод окремо дає помітне прискорення, найбільшого ефекту досягнуто при їх одночасному застосуванні. Час виконання запиту скоротився з 2831.869 мс до 0.076 мс, що становить збільшення швидкодії більш ніж у 37000 разів. Отримані дані підтверджують існування синергетичного ефекту та свідчать про високу практичну цінність комплексної оптимізації для систем, що працюють з великими аналітичними навантаженнями в умовах розподіленої обробки даних. Представлений підхід може бути використаний для модернізації існуючих інформаційних систем і підвищення їхньої ефективності при масштабуванні.

Ключові слова: *оптимізація запитів, розподілені бази даних, кешування, матеріалізація, індексація, PostgreSQL*

Problem statement

Optimizing query performance in distributed database management systems is one of the main challenges for modern information systems. With the rapid growth of data volumes and increasing demands on processing speed, traditional approaches to working with databases no longer provide the required level of efficiency. This is particularly noticeable when executing complex analytical queries that require complex and costly operations such as joining, sorting, or aggregating large data sets.

Distributed databases are characterized by highly complex architecture and the need for effective coordination between different system nodes. In contrast to centralized systems, distributed databases implement the storage and processing of information on a set of interconnected nodes that function as a single logical system, despite their geographical distance. This architecture creates additional challenges related to network delays, data synchronization, and load balancing. When executing complex queries involving numerous table joins, nested subqueries, and sorting operations, system response times can reach several seconds, which is unacceptable for most business applications. Therefore, developing and implementing effective query optimization techniques is critical to ensuring the performance of distributed database management systems.

The problem of query optimization is particularly relevant for analytical platforms that process multidimensional queries over large historical data sets. Such tasks are typical for financial, telecommunications, and trading systems, where millions of records need to be processed and complex calculations performed in real time. Without the use of effective optimization techniques, such queries can take hours to execute, effectively rendering them unusable in practice.

A comprehensive approach to optimisation, which combines different techniques, can achieve the best results. The most effective of these are caching, indexing, and materialisation of views. Caching decreases the processing time of repeated queries by storing the results in fast memory. The indexing creates auxiliary structures that reduce the number of disk accesses and speed up searches. Materialisation, in turn, involves pre-calculating complex results and storing them for future use.

Using caching can significantly reduce the load on the database, especially in systems with a high proportion of repetitive queries. Materializing intermediate calculation results avoids repeating complex operations, which is especially important for analytical systems. Creating appropriate indexes can speed up search and filter operations by several orders of magnitude, turning a full table scan into a quick index search.

Distributed query processing requires an effective planning, taking into account data location, network bandwidth, and the current load on nodes. Suboptimal planning can lead to an excessive amount of data transfer, which becomes a major performance limitation.

Consequently, research and comparative analysis of various methods for optimizing queries in distributed databases is relevant, as it will enable the identification of the most effective strategies. Of particular interest is the study of the synergistic effect of the combined application of various optimization methods. This research is of important practical significance for database developers and administrators, as it allows for improving the efficiency of systems that work with large volumes of information in a distributed environment.

Recent research and publications analysis

Modern scientific works demonstrate a constant growth of interest in the problem of query optimization in distributed database management systems. Many researchers focus on finding ways to reduce query execution time and improve resource utilization. The relevance of this topic is due to the continuous increase in the amount of information and the growing demands for its processing speed.

Research [1] demonstrates the importance of splitting complex queries to improve the performance of distributed databases. The authors show that decomposing queries into simpler components reduces the workload on individual system nodes and improves overall processing efficiency. Particular attention is paid to the analysis of distributed query execution algorithms and the minimization of inter-node data exchange. The researchers note that the effectiveness of this approach depends on the characteristics of the query, the volume of data, and the network topology, which determines the scalability and performance of the system.

The study [2] presents the QOTUM query optimizer designed for distributed databases in a cloud environment. The system uses a materialized view mechanism to reuse intermediate results and reduce query execution time. The results of the experiments showed a significant reduction in query execution time and memory usage. The authors also highlight the importance of considering cloud environment factors such as data distribution and network costs.

Sulima and Iermolaiev [3] examined approaches to optimizing SQL queries and demonstrated that even a slight change in the query structure can significantly affect its execution speed. The authors paid significant attention to the analysis of execution plans, the identification of "bottlenecks," and the detection of operations that cause delays during data processing. The study proposed an optimization method that involves replacing the IN operator with a temporary table with a non-clustered index. This approach reduces the number of logical data accesses and speeds up the selection process. The results of the experiments showed an average reduction in query execution time of 15–17%. The authors also emphasize the necessity of regular analysis and optimization of SQL queries, especially in cases of increasing information volumes and system load.

The publication [4] presents an overview of the main methods for optimizing distributed databases. The authors systematized approaches to improving performance, including indexing, caching, fragmentation, replication, and load balancing. The connection between database architecture and system performance is examined, and practical recommendations are provided for selecting the optimal strategy depending on the type of load and data volume.

The paper [5] provides a detailed analysis of various database indexing techniques and their impact on query performance. The researchers examine the features of B-tree, hash, text, geospatial, and composite indexes, describing their strengths and limitations in detail. It is noted that the choice of the appropriate index type is determined by the database structure, data amount, and query specifics. The issue of balancing the number of indexes with the cost of updating them during data changes is considered separately. The authors conclude that the reasonable use of indexes is an important factor in improving the performance of database management systems.

The research [6] presents an empirical analysis of query optimization strategies aimed at reducing execution latency in large-scale databases. The authors consider both traditional and AI-oriented approaches, including the use of machine learning methods for query cost estimation, adaptive query execution, and intermediate result caching. The experimental part covers SQL, NoSQL, and distributed cloud databases using TPC-H, TPC-DS, and real queries. The results show that the combination of indexing, adaptive execution, and AI planning reduces query execution time by 45–50%. The authors conclude that the efficiency of optimization is determined by the type of load (OLTP or OLAP) and the system architecture.

The study [7] examines best practices for optimizing SQL databases for working with large volumes of data. The authors focus on partitioning, sharding, caching, and database schema optimization techniques. The article provides practical recommendations on combining different optimization methods to ensure scalability and high performance of systems in a Big Data environment.

In summarizing the results of recent studies, it can be noted that the problem of query optimization in distributed databases remains relevant, and most scientific works focus on improving individual methods of increasing performance. At the same time, there is a lack of comprehensive approaches in scientific sources that combine methods to achieve a synergistic effect. Therefore, research aimed at developing and experimentally substantiating combined approaches to query optimization is of great scientific and practical importance.

Formulating the research goal

The purpose of the research is to compare methods for optimizing complex analytical queries in distributed databases, including evaluating the effectiveness of caching, materialization of views, indexing, and their combination to improve query processing performance.

Materials and results

Optimization of queries in distributed databases is based on the use of methods that reduce computational costs and the amount of memory accesses. The most common of these are caching, indexing, and materialization of views. Their combination allows for a synergistic effect, where the total performance increase exceeds the effect of using each method separately.

Caching is based on reusing the results of previously executed queries. When the same query is repeated, the system receives data directly from RAM, eliminating the need to reread it from the disk. This is particularly effective for frequently executed queries. The level of caching efficiency is determined by the ratio between the frequency of repeated requests and the available RAM, which is described by the cache hit ratio.

Indexing involves creating additional data structures that enable search and filtering operations to be performed with logarithmic rather than linear complexity. The research uses B-tree-based indexes, which guarantee ordered storage of keys and stable access times to the required records. Such structures are self-balancing because they automatically maintain the optimal tree height when adding or removing elements. As a result, a search that previously required a complete table scan is now performed in a few disk accesses, which is especially important when working with large historical data sets.

Materialization of views involves storing the results of complex calculations in separate tables, eliminating the need to repeat the same join or aggregation operations during subsequent queries. Such views store pre-calculated results on disk, and the optimization system automatically determines the feasibility of reusing them. Data relevance is maintained using full or incremental update mechanisms, depending on changes in the base tables.

The combination of caching, indexing, and materialization forms a multi-level optimization system in which indexes speed up searches, materialization reduces the amount of computation, and caching provides quick access to previously received results. This approach is particularly effective in distributed systems, where data transfer between nodes is a key factor affecting the overall query execution time.

The study was conducted using the company's database, which contains information about employees, their salaries, and departments. The system architecture consists of several interconnected tables: employee, salary, dept_emp, department, and others that show the structure of the organization. The data covers many years, which makes it possible to evaluate the effectiveness of optimization methods for large amounts of information. The system architecture diagram is shown in the fig. 1.

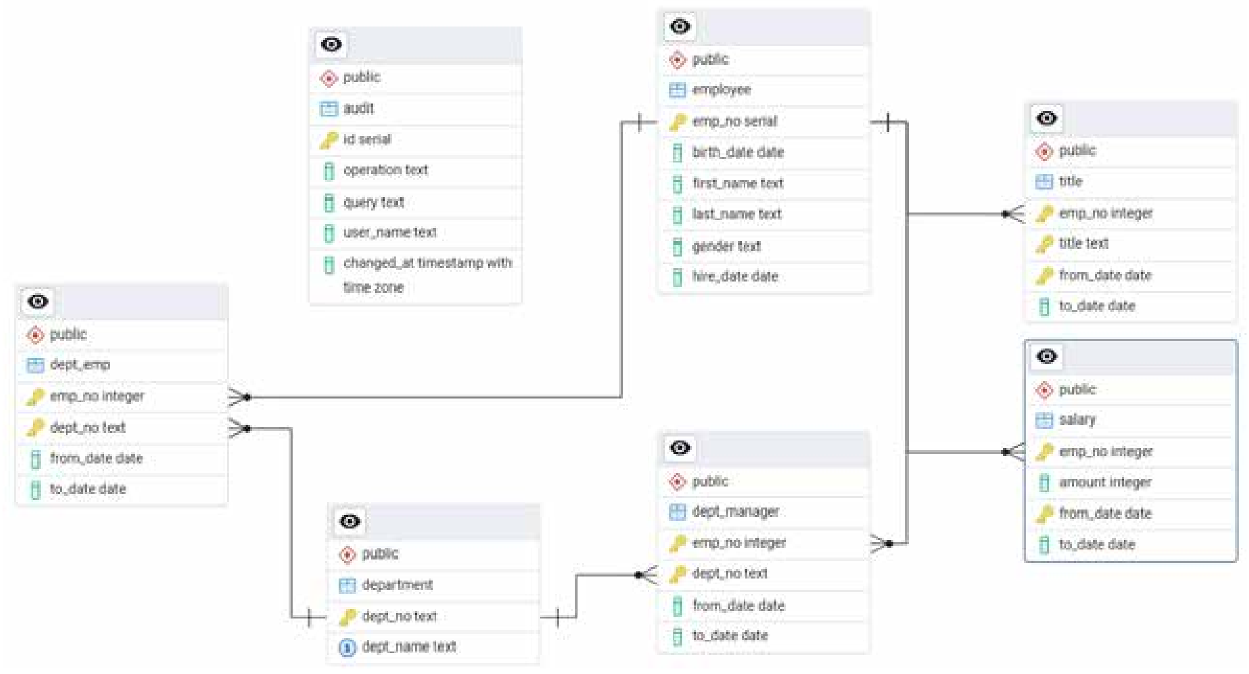

Fig. 1. Architecture of the experimental database

For the tests, we used an SQL query that combines several join and filter operations (Listing 1).

Listing 1 SQL query for conducting tests

```
WITH current_salaries AS (
  SELECT DISTINCT ON (emp_no) emp_no, salary
  FROM salary
  ORDER BY emp_no, from_date DESC
),
current_dept AS (
  SELECT DISTINCT ON (emp_no) emp_no, dept_no
  FROM dept_emp
  ORDER BY emp_no, from_date DESC
)
SELECT d.dept_name, e.emp_no, e.first_name, e.last_name, cs.salary
FROM employee e
JOIN current_salaries cs ON e.emp_no = cs.emp_no
JOIN current_dept cd ON cd.emp_no = e.emp_no
JOIN department d ON d.dept_no = cd.dept_no
WHERE e.hire_date <= DATE '2000-01-01'
ORDER BY d.dept_name, cs.salary DESC
LIMIT 100;
```

The query aggregates data from four tables, determines the current salary and department of each employee, filters the results by date of hire, and sorts them by salary and department. This structure allows us to accurately assess the impact of different optimization methods on execution time.

The results of the tests are shown in Table 1.

Table 1

Results of tests

№ of test	Optimization method	Description of conditions (cache/indexes/ materialization)	Planning Time (ms)	Execution Time (ms)
1	Without optimization (baseline)	After restarting PostgreSQL, without indexes, without MV	3.441	2831.869
2	Caching	Re-executing the same query	0.552	2612.267
3	Indexing	Executing the base query after creating indexes	0.453	1574.183
4	Materialization	Executing through a materialized view without indexes	0.077	83.989
5	Combination of methods	Materialization + indexes + caching	0.111	0.076

Analysis of the results showed that caching reduces query execution time by 7.8%, indexing by 44.4%, and materialization by 97%. The best result was achieved by combining all three methods, which reduced the execution time from 2831.869 ms to 0.076 ms, i.e., approximately 37,000 times. The data obtained confirms the presence of a pronounced synergistic effect: each individual method accelerates only a certain stage of processing, while their combined use optimizes the system comprehensively. Fig. 2 shows a comparison of query execution times for different approaches, clearly demonstrating the advantages of combined optimization.

Conclusions

The research confirmed the effectiveness of a comprehensive approach to optimizing queries in distributed databases. Experimental results showed that the combination of caching, indexing, and materialization significantly reduces the execution time of complex analytical queries from 2831.869 ms to 0.076 ms, i.e., an acceleration of more than 37,000 times.

Each of the methods considered has its own area of effectiveness. Caching reduces the execution time of repetitive queries by using previously obtained results. Indexing improves the performance of search and join operations by eliminating the need for full table scans. Materialization stores the results of complex calculations in separate structures, preventing them from being re-executed.

The highest performance was demonstrated by the combined application of these methods, which creates a multi-level optimization system and provides a synergistic effect. The results obtained prove that a systematic approach to query optimization can significantly reduce computational costs and increase the scalability of distributed systems.

The practical significance of the study is the possibility of applying the proposed methodology by database developers and administrators to improve the efficiency of information systems, in particular those that process large amounts of data and perform analytical tasks.

Further research could focus on an in-depth analysis of the effectiveness of other optimization methods, as well as on studying various combinations of these approaches to achieve an optimal balance between speed, resource utilization, and system stability.

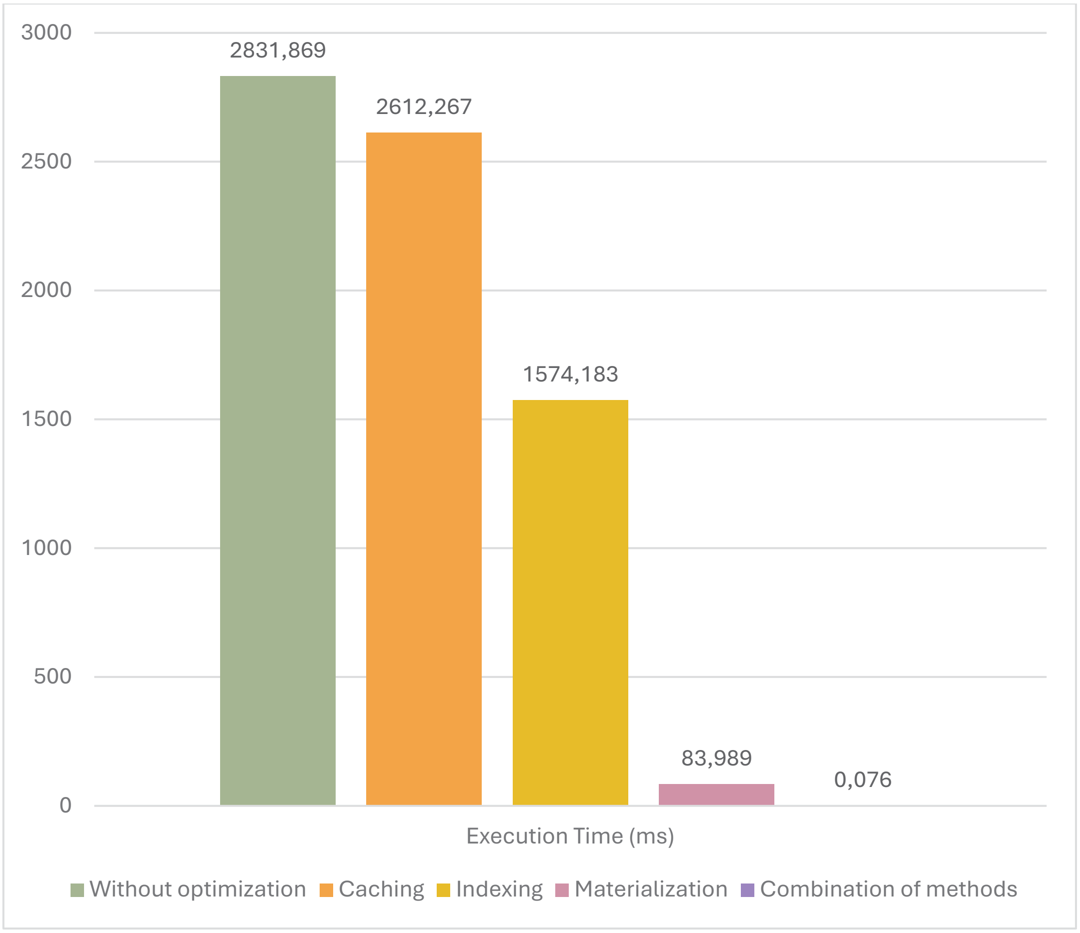

Fig. 2. Comparison of Execution Time for each method and for their combination

Bibliography

1. Панасюк В. В., Майданюк В. П. Оптимізація продуктивності розподілених баз даних через розділення складних запитів. *Матеріали LIV Всеукраїнської науково-технічної конференції підрозділів ВНТУ*. 2025. URL: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/48587/23928.pdf?sequence=3&isAllowed=y (дата звернення: 27.11.2025).

2. Archana B., Vilas K., Madhukar S. QOTUM: The Query Optimizer for Distributed Database in Cloud Environment. *TECHNICAL JOURNAL*. 2024. Т. 18, вип. 1. С. 172–177. ISSN ISSN 1846-6168 (Print), ISSN 1848-5588 (Online). URL: https://doi.org/10.31803/tg-20230501083155.

3. Суліма С. В., Єрмолаєв О. Д. Метод оптимізації SQL запитів системи управління базами даних. *Національний технічний університет України «КПІ імені Ігоря Сікорського»*. Київ, Україна, 2023. Т. 2, вип. 72 : Системи управління навігації та зв'язку Збірник наукових праць. С. 151–157. URL: https://doi.org/10.26906/SUNZ.2023.2.151.

4. Белоус Р., Крилов Є., Анікін В. Методи оптимізації запитів розподілених БД. *Міжвідомчий науково-технічний збірник*. 2021. Т. 2, вип. 39 : Адаптивні системи автоматичного управління. С. 3–11. ISSN 1560-8956. URL: https://doi.org/10.20535/1560-8956.39.2021.247364.

5. Голубінка В., Худий А. Підвищення продуктивності запитів до баз даних: аналіз технік індексації. *Вісник Національного університету "Львівська політехніка"*. Львів, Україна, 2024. вип. 15 : Інформаційні системи та мережі. С. 65–73. URL: https://doi.org/10.23939/sisn2024.15.065.

6. Stewart N., Adams D., Foster U. Empirical Analysis of Query Optimization Strategies for Reducing Execution Latency in Large-Scale Databases. *Journal of Innovation in Governance and business practices*. 2025. Т. 1, вип. 1. С. 88–117. URL: https://jigbp.com/index.php/jigbp/article/view/4 (дата звернення: 27.11.2025).

7. Uzzaman, A., Jim, M. M. I., Nishat, N., & Nahar, J. Optimizing SQL databases for big data workloads: techniques and best practices. *Academic journal on business administration, innovation & sustainability*. 2024. Т. 4, вип. 3. С. 15–29. ISSN 2997-9552. URL: https://www.researchgate.net/profile/Janifer-Nahar/publication/381725561_OPTIMIZING_SQL_DATABASES_FORBIG_DATA_WORKLOADS_TECHNIQUES_AND_BEST_PRACTICES/links/667fcab2f3b61c4e2c99919b/OPTIMIZING-SQL-DATABASES-FORBIG-DATA-WORKLOADS-TECHNIQUES-AND-BEST-PRACTICES.pdf (дата звернення: 27.11.2025).

References

1. Panasiuk, V. V., & Maidaniuk, V. P. (2025). Optymizatsiia produktyvnosti rozpodilenykh baz danykh cherez rozdilennia skladnykh zapytiv [Optimization of distributed database performance through complex query partitioning]. *Materialy LIV Vseukrayins'koyi naukovo-tekhnichnoyi konferentsiyi pidrozdiliv VNTU*. https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/48587/23928.pdf?sequence=3&isAllowed=y

2. Archana, B., Vilas, K., & Madhukar, S. (2024). QOTUM: The query optimizer for distributed database in cloud environment. *Technical Journal*, 18(1), 172–177. https://doi.org/10.31803/tg-20230501083155

3. Sulima, S. V., & Iermolaiev, O. D. (2023). Metod optymizatsii SQL zapytiv systemy upravlinnia bazamy danykh [Method for optimizing SQL queries in a database management system]. *Natsional'nyy tekhnichnyy universytet Ukrayiny «KPI imeni Ihorya Sikors'koho»*, 2(72), 151–157. https://doi.org/10.26906/SUNZ.2023.2.151

4. Belous, R., Krylov, Ye., & Anikin, V. (2021). Metody optymizatsii zapytiv rozpodilenykh BD [Methods of optimization of distributed databases]. *Mizhvidomchyy naukovo-tekhnichnyy zbirnyk*, 2(39), 3–11. https://doi.org/10.20535/1560-8956.39.2021.247364

5. Holubinka, V., & Khudyi, A. (2024). Pidvyshchennia produktyvnosti zapytiv do baz danykh: analiz tekhnik indeksatsii [Enhancing database query performance: Analysis of indexing techniques]. *Вісник Національного університету "Львівська політехніка"*, 15, 65–73. https://doi.org/10.23939/sisn2024.15.065

6. Stewart, N., Adams, D., & Foster, U. (2025). Empirical analysis of query optimization strategies for reducing execution latency in large-scale databases. *Journal of Innovation in Governance and Business Practices*, 1(1), 88–117. Retrieved November 27, 2025, from https://jigbp.com/index.php/jigbp/article/view/4

7. Uzzaman, A., Jim, M. M. I., Nishat, N., & Nahar, J. (2024). Optimizing SQL databases for big data workloads: Techniques and best practices. *Academic Journal on Business Administration, Innovation & Sustainability*, 4(3), 15–29. Retrieved November 27, 2025, from https://www.researchgate.net/profile/Janifer-Nahar/publication/381725561_OPTIMIZING_SQL_DATABASES_FORBIG_DATA_WORKLOADS_TECHNIQUES_AND_BEST_PRACTICES/links/667fcab2f3b61c4e2c99919b/OPTIMIZING-SQL-DATABASES-FORBIG-DATA-WORKLOADS-TECHNIQUES-AND-BEST-PRACTICES.pdf

Дата першого надходження рукопису до видання: 26.11.2025
Дата прийнятого до друку рукопису після рецензування: 15.12.2025
Дата публікації: 31.12.2025