Enhancing database performance through SQL optimization, parallel processing and GPU integration

Marat Nuriev[1*], *Rimma* Zaripova[2], *Alexey* Sinicin[2], *Andrey* Chupaev[3], and *Maksim* Shkinderov[1]

[1]Kazan National Research Technical University named after A. N. Tupolev – KAI, Kazan, Russia
[2]Kazan State Power Engineering University, Kazan, Russia
[3]Kazan National Research Technological University, Kazan, Russia

Abstract. This article delves into the cutting-edge methodologies revolutionizing database management systems (DBMS) through the lens of SQL query optimization, parallel processing, and the integration of graphics processing units (GPUs). As the digital world grapples with ever-increasing volumes of data, the efficiency, speed, and scalability of database systems have never been more critical. The first section of the article focuses on SQL query optimization, highlighting strategies to refine query performance and reduce resource consumption, thus enhancing application responsiveness and efficiency. The discourse then transitions to parallel processing in databases, an approach that leverages multiple processors or distributed systems to significantly boost data processing capabilities. This segment explores the advantages of parallelism in managing large datasets and complex operations, addressing the challenges and the impact on system scalability and fault tolerance. Furthermore, the article examines the innovative application of GPUs in database management, a development that offers profound speedups for analytical and machine learning tasks within DBMS. Despite the complexities and the initial investment required, the utilization of GPUs is portrayed as a game-changer in processing large-scale data, thanks to their highly parallel architecture and computational prowess. Together, these advancements signify a transformative shift in database technologies, promising to address the challenges of modern data management with unprecedented efficiency and scalability. This article not only elucidates these sophisticated technologies but also provides a glimpse into the future of database systems, where optimization, parallel processing, and GPU integration play pivotal roles in navigating the data-driven demands of the contemporary digital landscape.

1 Introduction

In the contemporary era, where information is a pivotal resource that supports our daily existence – from mundane activities to complex business operations – databases emerge as

* Corresponding author: marat_nu1@mail.ru

the cornerstone for storing and retrieving invaluable data. Amidst this data-driven landscape, SQL queries serve as the primary conduit for accessing desired information swiftly and efficiently [1,2]. This pivotal role underscores the critical importance of optimizing SQL queries to bolster application performance and efficiency. Slow queries not only frustrate users but also impede system performance and escalate operational expenses, thereby highlighting the necessity for effective SQL query optimization [3].

The imperative for optimizing SQL queries gains further prominence against the backdrop of escalating data volumes and the pressing demand for rapid data processing capabilities. This article embarks on an in-depth exploration of the myriad strategies and methodologies that can be employed to refine the performance of SQL queries within relational database management systems (RDBMS) [4,5]. Our discourse extends beyond the mere identification of optimization principles and approaches; it encompasses a thorough examination of common pitfalls and errors that detrimentally impact query performance.

The overarching aim of this scholarly piece is twofold: firstly, to underscore the criticality of rigorous SQL query optimization in achieving and sustaining superior application performance [6]; and secondly, to furnish a compendium of actionable insights, practical advice, and recommendations that can be leveraged by developers and database administrators in their professional endeavors [7,8]. By distilling the essence of these optimization techniques, the article seeks to empower practitioners with the knowledge to markedly enhance the responsiveness of their systems to user queries, alleviate server resource burdens, and by extension, amplify user satisfaction and optimize the efficiency of business processes [9].

As we traverse the subsequent sections, our narrative will meticulously dissect the mechanisms for diagnosing performance anomalies in SQL queries, spotlight the assortment of tools at our disposal for optimization purposes, and delineate the best practices that should be adopted to cultivate and maintain the health of relational databases. This comprehensive treatise on SQL query optimization is poised to serve as an indispensable resource for those seeking to navigate the complexities of database management in an increasingly data-intensive world.

2 Advanced techniques for optimizing table structure in relational databases

The structural optimization of tables within relational database management systems is a foundational strategy for enhancing the performance of SQL queries and, by extension, the overall efficiency and responsiveness of database-driven applications. This comprehensive exploration delves into sophisticated methodologies and best practices for refining table structures, aimed at achieving optimal performance and data integrity [10,11].

Comprehensive strategies for splitting large cables:

- Table decomposition and normalization: beyond merely splitting tables based on column count, table decomposition involves a thoughtful analysis of data relationships and usage patterns. Employing normalization principles (up to the third normal form or beyond, where applicable) ensures that data is stored without redundancy, facilitating more efficient updates and retrievals. This process often results in a schema where information is spread across multiple, focused tables, each serving a distinct aspect of the data model, thereby significantly reducing the data processed for individual queries [12].
- Vertical partitioning: this strategy involves separating a table into smaller, more manageable tables, each containing a subset of the original table's columns. This is particularly effective for tables with 'wide' structures – those with a large number of columns – where many columns are not frequently accessed together [13,14]. Vertical partitioning enhances performance by allowing queries to scan smaller datasets and by improving cache efficiency.

Optimized data type selection:
- Precision in data type choice: beyond minimizing data size, the precision in selecting data types involves understanding the nature of the data being stored and choosing types that most accurately represent it. For example, distinguishing between integer types based on their size (`TINYINT`, `SMALLINT`, `INT`, `BIGINT`) can have significant storage and performance implications, especially in large datasets [15].
- Temporal data types: for date and time information, choosing the correct granularity (e.g., `DATE`, `TIME`, `DATETIME`, `TIMESTAMP`) can optimize both storage and query performance. Avoiding over-precision in temporal data storage is key to balancing efficiency and functionality [16].
- Adoption of advanced data types: modern RDBMS offer advanced data types like `JSON`, `XML`, or spatial types, which can provide more efficient storage and querying mechanisms for complex data. Utilizing these advanced types where appropriate can leverage the database engine's optimizations for these formats.

Enforcing constraints for performance:
- Indexing strategies: while not a constraint per se, judicious use of indexes is vital. Primary and foreign keys inherently create indexes that assist in maintaining data integrity and improving query performance. Beyond these, consider indexing columns frequently used in `WHERE` clauses, `JOIN` conditions, or as part of an `ORDER BY`.
- Constraint-based optimization: utilizing constraints like `NOT NULL`, `UNIQUE`, and `CHECK` not only enforces data integrity but can also guide the database engine to make more efficient query execution plans by understanding the data distribution and inherent restrictions [17,18].

Efficient management of binary large objects (BLOBs):
- External storage for BLOBs: for large binary objects, such as images or documents, storing these outside the database and keeping a reference link or path in the table mitigates the performance impact on the database. This strategy utilizes the file system for what it's designed for – efficient file storage – while keeping the database agile and focused on transactional operations [19,20].
- Hybrid storage solutions: some modern databases support hybrid approaches, allowing for the storage of large objects in a way that can be efficiently managed and queried. Leveraging such features can offer the best of both worlds [21]: the performance and manageability of external storage with the transactional integrity and accessibility of database storage.

Implementation and continuous evaluation:
- Regular performance auditing: structural optimization is not a one-time task but a continuous process of evaluation and adjustment. Regularly reviewing the schema design, query performance, and system workload can uncover new opportunities for optimization.
- Adaptive schema design: as application requirements evolve, so too should the database schema. Adaptive design involves revisiting table structures, relationships, and data types in response to changing data access patterns, application features, and performance metrics.

Optimizing table structure in relational databases is a multifaceted endeavor that requires a deep understanding of both the data being managed and the capabilities of the database system. By implementing these advanced strategies [22,23], organizations can achieve significant improvements in database performance, scalability, and reliability, ensuring that their applications can meet the demands of today's data-intensive environments.

3 Advanced optimization techniques for Select queries in SQL

The optimization of SELECT queries is paramount for ensuring the high performance and scalability of database-driven applications. As SELECT queries often constitute the bulk of database operations, especially in data-intensive applications, their efficiency directly

influences the overall performance and user experience. This detailed exploration provides an in-depth look at sophisticated strategies for optimizing SELECT queries, focusing on reducing execution times and resource consumption [24,25].

Strategic data retrieval:

- Selective column fetching: explicitly specifying the required columns in a SELECT statement, as opposed to using `SELECT *`, minimizes the data load, reducing both CPU and I/O overhead. This practice is especially beneficial in tables with wide rows or numerous columns, where fetching unnecessary data can significantly impact performance [26,27].
- Row limitation techniques: for interfaces displaying paginated data or when only a subset of records is needed, employing `LIMIT` (or `TOP` in some RDBMS) and `OFFSET` clauses can drastically decrease the amount of data transferred, processed, and rendered. This approach is critical for enhancing responsiveness in applications dealing with large datasets.
- Efficient filtering with WHERE clauses: crafting precise `WHERE` clauses that effectively narrow down the result set can lead to substantial performance gains. Utilizing indexed columns within these clauses can further expedite data retrieval by allowing the database engine to quickly locate relevant rows [28,29].

Mastering JOINs for efficiency:

- Index-backed JOINs: ensuring that all columns used in JOIN conditions are indexed is crucial. Indexes facilitate rapid lookups and data association between tables, significantly speeding up JOIN operations.
- Data type alignment in JOINs: matching the data types of columns involved in JOINs eliminates the need for implicit type conversion, which can be a silent performance killer. Consistency in data types leads to more efficient comparisons and faster JOIN execution.
- Avoiding JOINs on low-cardinality columns: JOIN operations on columns with a limited range of unique values (low cardinality) can lead to inefficient execution plans, including full table scans. Prioritizing high-cardinality columns for JOIN conditions can enhance the efficiency of these operations [30].
- Considerations for data denormalization: in scenarios where the performance impact of frequent JOINs outweighs the benefits of normalization, selectively denormalizing the data schema might offer a beneficial trade-off. This involves strategically duplicating certain data across tables to reduce or eliminate the need for JOINs, thereby simplifying queries and speeding up data retrieval [31].

Caching strategies for SELECT queries:

- Result set caching: implementing caching mechanisms for frequently accessed data or query results can dramatically reduce database load by serving repeated requests from memory, bypassing the need for query re-execution [32,33]. Application-level caching, distributed caching systems (like Redis or Memcached), or even RDBMS-specific query caching features can be employed to achieve this.
- Precomputed aggregates: for data that involves complex calculations or aggregations, storing precomputed results in separate tables or materialized views can offer instant access to such information, eliminating the need for on-the-fly computation.

WHERE clause optimization:

- Condition ordering and index utilization: organizing conditions in a `WHERE` clause to exploit indexes effectively and placing the most selective conditions first can reduce the data set size early in the query execution process [34]. This leads to more efficient use of indexes and quicker retrievals.
- Using `IN` versus `OR`: the `IN` operator is generally more optimized than equivalent `OR` conditions, leading to clearer syntax and potentially faster execution [35].
- Careful use of wildcards with `LIKE`: when using the `LIKE` operator, leading wildcards (e.g., `%search`) can prevent the use of indexes, degrading performance. Whenever possible,

avoid leading wildcards or consider full-text search capabilities for more efficient text-based searches.

Enhancements in sorting and grouping:

- Sorting optimization: when sorting data (`ORDER BY`), consider the use of indexed columns to leverage the database's ability to quickly organize the data. Sorting by primary key or indexed columns can be much faster than sorting by non-indexed fields [36].
- Streamlined grouping operations: minimizing the number of columns used in `GROUP BY` clauses reduces computational overhead. When aggregation functions are used, filtering data with `WHERE` before grouping it with `GROUP BY` is more efficient than filtering after aggregation with `HAVING`.
- Avoidance of complex nested groupings: while SQL allows for nested `SELECT` statements and complex grouping, these constructs can lead to significant processing overhead. Simplifying query structures and avoiding unnecessary nested groupings can improve performance [37].

Adhering to these advanced optimization techniques ensures that SELECT queries are executed with maximum efficiency, leading to faster response times and a better user experience [38]. Continuous monitoring, analysis, and refinement of query performance are essential practices for database professionals aiming to optimize SQL query operations fully.

4 Advancements in parallel processing for database management systems

The exponential growth of data and the burgeoning demand for real-time processing and analytics have propelled a transition from traditional serial database management techniques to advanced parallel processing methodologies. Parallel processing in databases harnesses multiple processing units to execute tasks simultaneously, markedly enhancing data throughput and processing speed. This comprehensive examination delves into the nuanced facets of parallel processing within database management systems, shedding light on strategies, advantages, challenges, and the forward path in harnessing these technologies for sophisticated data management solutions [39,40].

At the heart of parallel processing in the context of databases is the concept where database operations are executed concurrently across multiple processors or across a network of interconnected computers. This approach encompasses various strategies, including data partitioning, parallel query processing, parallel transaction processing, and distributed computing [41]. Data partitioning breaks down a large database into smaller segments that can be processed independently, enabling efficient query processing and data management. Parallel query processing involves dissecting complex queries into smaller sub-queries that are executed across different processors, thereby reducing overall execution time. Parallel transaction processing enhances throughput and minimizes latency, necessitating sophisticated concurrency control mechanisms to maintain transaction integrity and consistency. Distributed computing, utilizing a network of interconnected computers, allows for scalable and efficient data processing on a grand scale [42].

The realization of parallel processing in databases is supported by advancements in hardware and software technologies, including modern CPUs with multiple cores, clusters of servers or distributed systems, and cloud-based platforms. These technologies enable parallel execution of processes, facilitating horizontal scaling and offering scalable and flexible resources for implementing parallel database processing.

However, implementing parallel processing in database systems presents challenges, including the complexity of system design and maintenance [43], the need for robust concurrency control and synchronization mechanisms to ensure data integrity, and the increased costs associated with infrastructure and energy consumption.

Looking forward, the future of parallel processing in database management is tied to technological advancements such as in-memory computing, GPU and Field Programmable Gate Arrays (FPGAs) acceleration, and autonomous databases. In-memory computing reduces access times, making real-time analytics feasible on a large scale. GPUs and FPGAs offer massive parallel processing capabilities for accelerating complex computations, while emerging technologies aim to automate optimization [44], scaling, and maintenance tasks in parallel processing systems.

Parallel processing represents a paradigm shift in database management, promising unprecedented levels of efficiency, scalability, and performance. Through strategic implementation and the leveraging of cutting-edge technologies, parallel processing in databases stands as a pivotal solution to the challenges posed by the modern data landscape. The continuous evolution of hardware capabilities and the advent of smarter, autonomous systems are expected to further enhance the potential of parallel processing in managing the deluge of data, marking a new era in database technology [45,46].

5 Advantages of using GPUs in database management

The utilization of graphics processing units for database processing marks a significant paradigm shift in the realms of data management and analysis (Fig.1). Historically, GPUs were predominantly associated with rendering graphics in video games and other visual applications. However, their highly parallel architecture has rendered them powerful accelerators for database operations, addressing the demands of today's data-intensive applications with unprecedented efficiency [47]. This exploration delves into how FPGAs, with their capacity to execute thousands of threads simultaneously, stand out as ideal tools for handling the massive parallelism required in processing large datasets and executing complex analytical operations [48].

Unlike central processing units (CPUs) that are designed with a limited number of cores for sequential processing and can handle a broad range of computing tasks, GPUs boast hundreds or thousands of smaller cores. These are specifically engineered for parallel processing of complex mathematical calculations, laying the foundation for their significant speedups in database operations, particularly for tasks that benefit from parallel execution such as data querying [49], aggregation, and machine learning model training on voluminous datasets.

The application of GPUs in database management introduces accelerated query processing, enabling dramatic reductions in the time required to perform data analysis and querying. By offloading intricate query operations to the GPU, databases can tap into the parallel processing prowess of GPUs to conduct multiple operations concurrently, thereby slashing query response times. Moreover, the computational might of GPUs finds a natural fit in data analytics and machine learning applications where processing large volumes of data swiftly is paramount. Such acceleration facilitates the rapid training of machine learning models, permitting more complex explorations and quicker iterations. Furthermore, GPUs emerge as more energy-efficient alternatives to CPUs for specific database tasks, courtesy of their higher computations per watt of power consumed, presenting a viable strategy to curb the energy footprint of data centers [50].

Nonetheless, the integration of GPUs into database systems is accompanied by hurdles, including the sophistication of hardware and software required to allocate and manage tasks across the GPU’s multitude of cores. This complexity calls for significant investments in both development and infrastructure. Another challenge lies in the latency introduced by data transfers between the main memory and GPU memory, especially notable with large datasets where optimizing data transfer and minimizing movement are crucial for harnessing GPU acceleration fully. Additionally, the development and optimization of applications to

leverage GPU capabilities demand specialized knowledge and tools, with the ecosystem for GPU-accelerated database operations still in need of further maturity and standardization.

Despite these challenges, the transformative impact of GPUs on database management is undeniable, propelling the field towards unprecedented performance levels for data-intensive applications. Industries where rapid data analysis is crucial, such as finance, e-commerce, and scientific research, stand to benefit significantly. With the continuous evolution of technology and tools supporting GPU-accelerated database processes, their adoption is poised to broaden, catalyzing innovations across data analytics, machine learning, and beyond.

The application of GPUs in database processing heralds a new era in data management, characterized by significant performance and efficiency gains. As the quest to leverage vast amounts of real-time data intensifies, GPUs are set to play an increasingly central role in database systems, representing a significant stride in the advancement of database technologies.

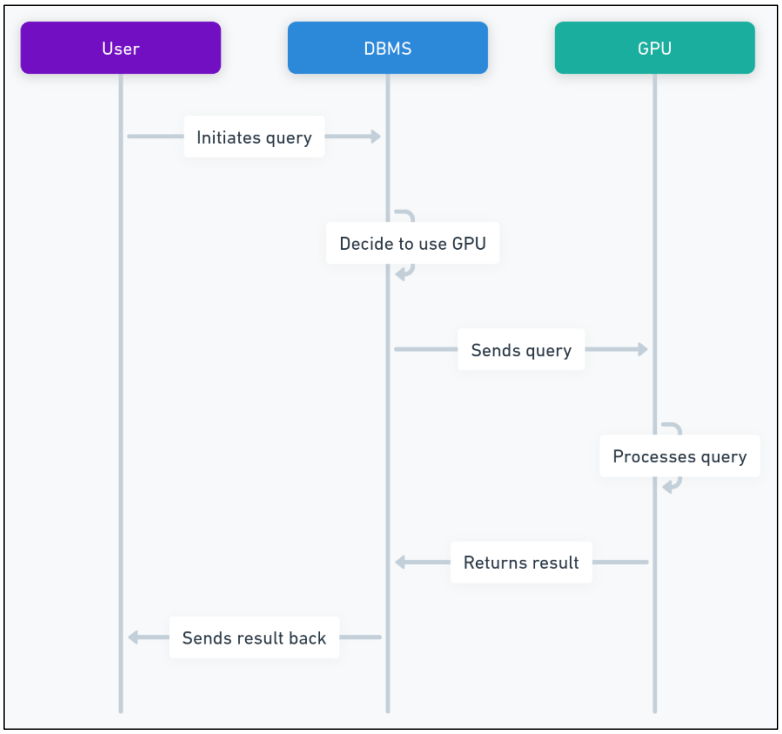

Fig.1. GPU acceleration in database processing.

6 Conclusion

In conclusion, the exploration into the realms of SQL query optimization, the strategic application of parallel processing approaches, and the innovative utilization of GPUs for database management collectively underscore a transformative era in database technologies. These advancements not only promise to elevate the efficiency, speed, and scalability of data management systems but also pave the way for new paradigms in handling the ever-expanding volumes of data in today's digital landscape.

Optimizing SQL queries, as discussed, is foundational for enhancing application performance and user satisfaction. By meticulously refining query structures and leveraging

best practices, developers can significantly reduce execution times and resource consumption, thereby ensuring more responsive and robust database systems.

Parallel processing, with its ability to distribute tasks across multiple processors or nodes, offers a compelling solution to the challenges posed by large-scale data processing and complex computational demands. This approach not only enhances performance and scalability but also introduces greater fault tolerance and efficiency in data management operations [51].

The integration of GPUs into database management systems marks a notable advancement, bringing substantial speedups in data processing tasks, particularly those involving complex analytics and machine learning models. Despite the challenges associated with their adoption, GPUs stand out for their unparalleled computational power and energy efficiency, heralding a new frontier in database processing capabilities.

As we stand on the brink of these technological advancements, it is clear that the future of database management is bright, with SQL optimization, parallel processing, and GPU utilization leading the charge. These strategies and technologies collectively offer the potential to tackle the burgeoning data challenges of the modern era, enabling faster, more efficient, and scalable database systems. The continuous evolution of these technologies, coupled with the growing expertise in their application, promises to unlock even greater potentials in data management, analytics, and beyond, setting the stage for an exciting future in database technology.

References

1. P. Ding, F. Wang, D. Gu, H. Zhou, Q. Gao, X. Xiang, 2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Tianjin, China, 1351-1355 (2018)
2. Z. Xu, H. Li, Y. Chen, S. Liu, Z. Wan, 2023 6th International Conference on Electronics Technology (ICET), Chengdu, China, 1156-1160 (2023)
3. J. Zimon, M. Zoworka, 2013 International Symposium on Electrodynamic and Mechatronic Systems (SELM), Opole-Zawiercie, Poland, 77-78 (2013)
4. A. P. Mudrov, F. F. Khabibullin, G. V. Pikmullin, Z. D. Gurgenidze, BIO Web of Conferences **52**, 00046 (2023)
5. M. G. Yarullin, F. F. Khabibullin, Lecture Notes in Mechanical Engineering, 145-153 (2017)
6. Y. Smirnov, A. Kalyashina, R. Zaripova, International Russian Automation Conference (RusAutoCon), Sochi, Russian Federation, 913-917 (2022)
7. Z. Gizatullin, R. Gizatullin, 2023 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, Russian Federation, 261-265 (2023)
8. Z. M. Gizatullin, M. P. Shleimovich, Russ. Aeronaut **66**, 154-161 (2023)
9. S. Lyasheva, R. Safina, M. Shleymovich, 2023 International Conference on Industrial Engineering, Applications and Manufacturing, 797-802 (2023)
10. M. Shleymovich, R. Safina, 2022 International Russian Automation Conference, 289-293 (2022)
11. R. M. Shakirzyanov, A. A. Shakirzyanova, 2021 International Russian Automation Conference (RusAutoCon), 714-718 (2021)
12. Y. I. Soluyanov, A. I. Fedotov, D. Y. Soluyanov, A. R. Akhmetshin, IOP Conference Series: Materials Science and Engineering **860(1)**, 012026 (2020)

13. A. V. Chupaev, R. S. Zaripova, R. R. Galyamov, A. Y. Sharifullina, E3S Web of Conferences **124**, 03013 (2019)
14. L. V. Plotnikova, R. R. Giniyatov, S. Y. Sitnikov, M. A. Fedorov, R. S. Zaripova, IOP Conference Series: Earth and Environmental Science **288**, 012069 (2019)
15. M. Tyurina, A. Porunov, A. Nikitin, R. Zaripova, G. Khamatgaleeva, Lecture Notes in Mechanical Engineering, 391-402 (2022)
16. E. I. Gracheva, O. V. Naumov, Journal of Pharmacy and Technology **8**, 4, 26763-26770 (2016)
17. D. D. Micu, I. V. Ivshin, E. I. Gracheva, O. V. Naumov, A. N. Gorlov, E3S Web of Conferences **124**, 02013 (2019)
18. O. Soloveva, S. Solovev, R. Zaripova, F. Khamidullina, M. Tyurina, E3S Web of Conferences **258**, 11010 (2021)
19. R. F. Gibadullin, N. S. Marushkai, 2021 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 404-409 (2021)
20. V. O. Kozelkova, G. A. Ovseenko, V. I. Karachin, T. Van Tung, N. C. Kien, R. S. Kashaev, 4th International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE), 1-4 (2022)
21. R. F. Gibadullin, I. S. Vershinin, R. Sh. Minyazev, 2017 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 1-6 (2017)
22. T. Petrov, A. Safin, E3S Web of Conferences **178**, 01049 (2020)
23. V. O. Kozelkova, G. A. Ovseenko, V. I. Karachin, N. C. Kien, T. Van Tung, O. V. Kozelkov, 4th International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE), 1-5 (2022)
24. R. F. Gibadullin, G. A. Baimukhametova, M. Yu. Perukhin, 2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 1-7 (2019)
25. G. R. Rakhmatullina, E. A. Pankova, O. V. Fukina, M. Khayytov, L. V. Chapaeva, Journal of Physics: Conference Series **2270(1)**, 012056 (2022)
26. V. A. Gerasimov, M. G. Nuriev, D. A. Gashigullin, 2022 International Russian Automation Conference (RusAutoCon), 75-79 (2022)
27. S. R. Khasanov, E. I. Gracheva, M. I. Toshkhodzhaeva, S. T. Dadabaev, D. S. Mirkhalikova, E3S Web of Conferences **178**, 01051 (2020)
28. Z. M. Gizatullin, R. M. Gizatullin, M. G. Nuriev, 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 120-123 (2020)
29. S. Lyasheva, M. Shleymovich, R. Shakirzyanov, 2019 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon), 1-6 (2019)
30. E. Gracheva, M. Toshkhodzhaeva, O. Rahimov, S. Dadabaev, D. Mirkhalikova, S. Ilyashenko, V. Frolov, International Journal of Technology **11**, 8 (2020)
31. M. Shakirzyanov, R. Gibadullin, M. Nuriyev, E3S Web of Conferences **419**, 02029 (2023)
32. T. Petrov, A. Safin, E3S Web of Conferences **178**, 01016 (2020)
33. J. Yoqubjonov, R. Gibadullin, M. Nuriev, E3S Web of Conferences **431**, 07011 (2023)
34. I. Viktorov, R. Gibadullin, E3S Web of Conferences **431**, 05012 (2023)
35. R. F. Gibadullin, I. S. Vershinin, M. M. Volkova, 2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon), 1-7 (2020)

36. R. F. Gibadullin, M. Yu. Perukhin, B. I. Mullayanov, 2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon), 1-6 (2020)
37. S. N. Cherny, R. F. Gibadullin, 2022 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 965-970 (2022)
38. V. A. Raikhlin, I. S. Vershinin, R. F. Gibadullin, Journal of Physics: Conference Series **2096**, 012160 (2021)
39. R. F. Gibadullin, I. S. Vershinin, R. Sh. Minyazev, 2018 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 1-6 (2018)
40. V. A. Raikhlin, R. F. Gibadullin, I. S. Vershinin, Lobachevskii Journal of Mathematics **43**, 2, 455-462 (2022)
41. G. A. Ovseenko, R. S. Kashaev, O. V. Kozelkov, T. K. Filimonova, T. S. Evdokimova, A. M. Mardanova, 5th International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE) **5**, 1-5 (2023)
42. R. Zaripova, A. Nikitin, Y. Hadiullina, E. Pokaninova, M. Kuznetsov, E3S Web of Conferences **288**, 01072 (2021)
43. I. N. Madyshev, V. V. Kharkov, N. Z. Dubkova, M. G. Kuznetsov, AIP Conference Proceedings **2647**, 1 (2022)
44. Z. M. Gizatullin, M. S. Shkinderov, R. R. Mubarakov, Proceedings of the 2022 Conference of Russian Young Researchers in Electrical and Electronic Engineering, 1350-1353 (2022)
45. Z. Gizatullin, M. Shkinderov, 2019 International Russian Automation Conference, 8867761 (2022)
46. A. G. Ilyin, A. S. Mahdi Khafaga, V. Yunusova, 2021 Systems of Signals Generating and Processing in the Field of on Board Communications, 1-4 (2021)
47. I. Barkov, C. S. Gabdrakhmanova, G. I. Gaptullazyanova, A. V. Kholkin, In Computer Applications for Management and Sustainable Development of Production and Industry (CMSD2021) **12251**, 26-35 (2021)
48. E. Kozlov, R. Gibadullin, E3S Web of Conferences **474**, 02031 (2024)
49. R. M. Petrova, E. Gracheva, 2023 5th International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA), Lipetsk, Russian Federation, 1049-1055 (2023)
50. R. M. Petrova, E. Gracheva, 2023 5th International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA), Lipetsk, Russian Federation, 1056-1061 (2023)
51. R. Zaripova, V. Kosulin, M. Shkinderov, I. Rakhmatullin, E3S Web of Conferences **460**, 04011 (2023)