European Journal of Technical and Natural Sciences 2026, No 4
ISSN 2414-2352

PREMIER Publishing
ppublishing.org

DOI:10.29013/EJTNS-26-4-82-90

AN ARCHITECTURAL MODEL FOR ENSURING FINANCIAL ACCURACY AND CONSISTENCY OF DATA IN DISTRIBUTED OPERATING SYSTEMS

Mikhaylov Bogdan [1], ***Omorov Akzholbek*** [2], ***Izripov Iusup*** [3]

[1] Senior Software Developer, VK, Moscow, Russia

[2] Senior Software Developer, John Deere, USA

[3] Senior Software Engineer, T-Bank (formerly Tinkoff), Russia

Cite: *Mikhaylov B., Omorov A., Izripov I. (2026). An architectural model for ensuring financial accuracy and consistency of data in distributed operating systems. European Journal of Technical and Natural Sciences 2026, No 4.* https://doi.org/10.29013/EJTNS-26-4-82-90

Abstract

An architectural model for ensuring financial accuracy and consistency in distributed operating systems is proposed. This model addresses the main reasons for discrepancies in financial transactions, such as parallel modifications of records, retransmissions of requests, differences in data formats, and failures of individual components. The architecture requirements include transaction integrity, processing sequence, and traceability of changes, fault tolerance, and information recovery. The main components of the model include a transaction loop, financial journal, event repository, message transmission mechanism, accounting subsystem, analytical subsystem, reconciliation tools, and control tools. Mechanisms such as unique identification of operations, idempotent processing, record versioning, and compensation for previous actions are also considered in this model. It has been found that the combination of these mechanisms can reduce the likelihood of duplicate operations, partial data changes, and conflicts between the components of a distributed system.

Keywords: *financial accuracy, data consistency, distributed system, financial transaction, transaction, event log, idempotent processing, data reconciliation, fault tolerance, data recovery*

Relevance of the study

Modern financial transactions are conducted through information systems that are distributed across various servers, data centers, and software services. These transactions can be reflected in multiple subsystems, including payment, accounting, analysis, and reporting. Therefore, the accuracy and consistency of financial information depend on the reliability of the calculations and data in all parts of the system (Kravchenko D. A., 2022, p. 44).

This issue is of particular significance for financial institutions, as errors, delays in information transmission, reprocessing of transactions, and inconsistencies in re-

cord modifications can lead to misrepresentation of information regarding payments, accounts, and liabilities. The requirements for the accuracy, completeness, timeliness, and comparability of data are established in international recommendations for financial risk management and reporting.

A distributed architecture makes it more challenging to maintain a unified data state, particularly in the event of communication failures, component failures, and concurrent processing of operations. Consequently, a model that integrates the mechanisms for transaction processing, synchronization, logging, reconciliation, and data recovery is necessary. Developing such a model would enhance the dependability of financial systems, minimize the risk of mistakes, and generate reliable accounting, managerial, and analytical data.

The purpose of the study

The aim of this research is to create an architectural model that guarantees the accuracy of financial transactions, consistent data modifications, and the maintenance of a single financial state across all components of a distributed system.

Materials and research methods

The research materials included open scientific, regulatory, methodological, and technical sources related to transaction processing, the quality of financial data, transaction logging, and the stability of information systems (Gorbunova A. V., 2025; Kravchenko D. A., 2022; Kuznetsov S. D., Poskonin A. V., 2013; Malyuga K. V., Perl I. A., Slapoguzov A. P., 2021; Popov S. G., Fridman V. S., 2018; Soloviev A. V., 2025; Sukhoplyuev D. I., Nazarov A. N., 2024; Tatarnikova T. M., 2023; Tatarnikova T. M., Shelest M. N., 2022; Fomin D. S., Balsamov A. V., 2021).

Methods such as analysis, generalization of sources, systematization of architectural requirements, and comparative analysis of distributed processing mechanisms were used. Additionally, structural and functional modeling was employed in the work.

The results of the study

Financial accuracy requires accurate representation of monetary values and the absence of errors when performing calculations. To store amounts, you must use precise numeric formats with a fixed number of decimal places. The use of approximate numeric formats can lead to inaccuracies, therefore, the rules for representing monetary amounts, currencies, rounding and bit depth should be the same for all components of the information system (Cowlishaw M. F., Schwarz E. M., Smith R. M., Webb C. F., 2001).

Data consistency at the transaction level means that simultaneous execution of multiple operations should not result in an outcome that would be impossible if the operations were performed sequentially. To achieve this, transaction isolation mechanisms are used to prevent incorrect modifications of interrelated records. In case of a conflict, one of the transactions can be canceled and re-executed. This approach is particularly important when modifying cash balances, limits, liabilities, and other financial indicators.

In a distributed system, it is essential to have a single procedure for recording transactions. Confirmed transactions should be counted in the order they were actually completed, in order to prevent conflicting information from being stored on different nodes of the system at the same time. Therefore, the architecture of a financial system must ensure the consistent confirmation of transactions, the synchronization of data copies, and a unified process for changing the financial status.

The main reasons for the decline in the quality of financial data are fragmented information infrastructure, outdated software solutions and a large number of manual operations. Due to the lack of a unified system for classifying data and information about their origin, it is difficult to combine data, identify errors, and determine when a discrepancy occurred (Kuznetsov S. D., Poskonin A. V., 2013, p. 340).

Inconsistencies can arise from differences in names, identifiers, grouping rules, and how information is presented. If information about accounts, clients, counterparties, or transactions is stored in different systems according to different rules, combining it may result in inaccurate results. To prevent these discrepancies, it is essential to use uniform reference books, identifiers, naming conventions, and automated reconciliation processes.

The architectural model for ensuring financial accuracy and consistency of data should cover all stages of information processing, from the registration of a financial transaction to its reflection in the accounting, settlement, and analytical subsystems. The main requirements for this model are related to computational accuracy, transaction integrity, data uniformity, fault tolerance, and the ability to monitor and restore information (Soloviev A. V., 2025, p. 115). These requirements are presented in (table 1).

Table 1. *Basic requirements for an architectural model to ensure financial accuracy and consistency of data*

Requirement	Content of the requirement
The accuracy of financial calculations	The use of precise numerical types, uniform rounding rules, a set bit depth, and consistent processing order for monetary values in all components of the system
Transaction integrity	Execution of related changes in a single transaction, with confirmation of each operation or cancellation of the entire transaction
Processing sequence	Establishing a unified procedure for recording transactions and implementing transaction isolation mechanisms
Safe re-execution	Repeated processing of cancelled or unconfirmed transactions should not result in repeated write-offs, accruals, or changes to the financial results
Uniformity of data	The use of general reference books, classifiers, formats, names, and identifiers for objects
Traceability of data origin	Storing information about the original source of the information, the steps taken to transform it, and any changes made
Automated reconciliation	Data comparison between operational, accounting, settlement, and analytical systems
Fault tolerance	Maintaining the essential functions of the system in the event of communication failure, increased workload, or failure of individual components
Registration of actions	Maintaining secure logs that contain information about the time, source, action, and result of each event
Protecting information integrity	Restrictions on unauthorized modification of financial data and transaction logs
Backup	Regularly create backups of critical data and verify their ability to be recovered
Crash Recovery	There is a set process for restoring data and verifying its integrity after resuming operations
Responsibility for data quality	Consolidation of data control powers and responsibilities at all stages of data creation, processing, and storage

A source: author's development based on (Kuznetsov S. D., Poskonin A. V., 2013; Soloviev A. V., 2025; Sukhoplyuev D. I., Nazarov A. N., 2024; Fomin D. S., Balsamov A. V., 2021)

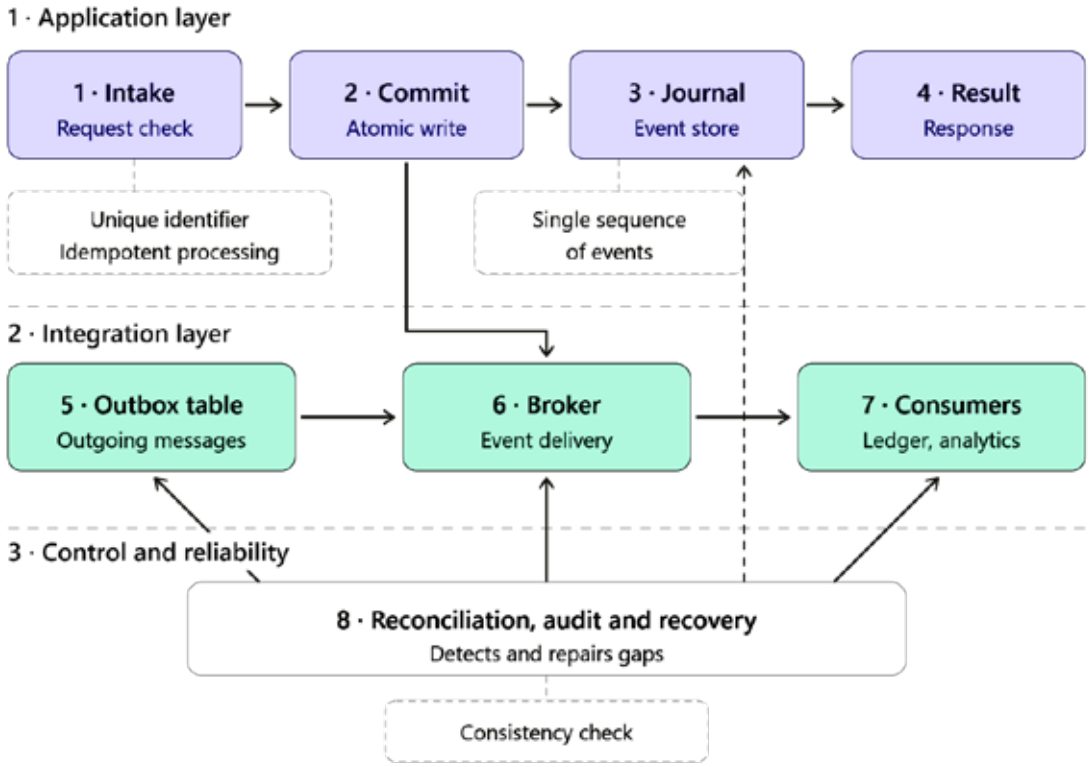
The proposed architectural model revolves around a single framework for recording financial transactions. Initially, the system receives a request and assigns a unique identifier to it. The transaction is then verified and executed in the processing module. The outcome is recorded in a financial ledger, which maintains a consistent record of changes. Simultaneously, information about the approved transaction is added to the outgoing message queue, from where it is transmitted to other system components.

This scheme reduces the gap between changing data and sending information. Based on the outgoing message queuing mechanism, a financial transaction and its corresponding event are stored within a single transaction. If the process is canceled, the message is not sent. However, if the transaction is approved, a separate process passes the event to the message broker (Amazon Web Services).

The financial journal serves as the primary source of information regarding changes made. By storing events in the order they occur, you can restore the state of an object at a specific moment, track the sequence of operations, and generate data for various subsystems. However, derivative representations may not always be updated in real-time, so they should not be relied upon as the sole source of confirmation for a financial transaction.

The model consists of a main transaction loop, financial record log, event transmission, accounting, and analytical representations, as well as a reconciliation module (Figure).

Figure 1. *Architectural model for ensuring financial accuracy and consistency of data (developed by the author)*



To implement the model, various mechanisms are used, each of which handles a specific distributed processing task. (Table 2)

Table 2. *Mechanisms for the implementation of the architectural model*

Mechanism	Appointment
The unique identifier of the operation	Recognizing a repeated request
Idempotent processing	Avoiding repeated data changes during message retransmission
Transactional outgoing message table	Joint recording of financial record and event
Sequential event log	Saving the history and order of changes
Versioning records	Detecting simultaneous changes to a single object
Serializable isolation	Exclusion of results that are impossible with sequential processing

Mechanism	Appointment
Compensating operation	Correction of previously performed actions in case of failure of the next stage
Control reconciliation	Comparison of the operation log and derived representations

A source: author's development based on (Malyuga K. V., Perl I. A., Slapoguzov A. P., 2021; Soloviev A. V., 2025; Fomin D. S., Balsamov A. V., 2021)

The initial conditions for formalizing the model include the equality of interrelated financial records, the absence of repeated changes in the result when processing an operation again, and a consistent sequence of transactions. These conditions are then supplemented by the requirements for accurate representation of monetary values and control verification of the overall journal position.

Each operation must be reflected completely and accurately, with no discrepancies between related records. Repeated receipt of a previously processed request should not result in repeated debit, charge, or balance change. Simultaneous transactions should produce the same result when processed in sequence.

The model also requires that subsystems maintain a causal order of confirmed transactions, which can be brought together in a single financial log. This ensures accurate calculations, eliminates duplicate entries, and allows for a unified financial state of the distributed system.

From a philosophical and methodological point of view, it is incorrect to equate financial accuracy in a distributed system with the equal distribution of all physical copies of data. Due to the fact that the transmission of messages and the creation of analytical representations require limited time, individual subsystems may be at different stages of processing the same data. A more accurate criterion for financial reliability is the preservation of financial invariants within an authoritative transaction cycle, as well as the possibility of comparing each derivative representation with the same confirmed journal entry (Lamport L., 1978; Bailis P., Fekete A., Franklin M. J., Ghodsi A., Hellerstein J. M., Stoica I., 2014; Bação J. L., Guerreiro S., 2026).

With this approach, the financial journal records the operational fact as recognized by the system. Accounting, reporting, and analytical subsystems provide different projections of this fact. However, the formal consistency of records does not necessarily ensure the economic reliability of an original event. An erroneous amount, incorrect invoice, or operation without proper authorization may also be technically consistent with the records. Therefore, it is important to apply mathematical conditions in conjunction with verifying banking details, credentials, and data origin to ensure accuracy and reliability.

Let c denote the currency, o denote the financial transaction, S denote the state of the system, $T(o)$ denote the transformation of the state by operation o, J denote the sequential log of confirmed transactions, and w denote the overall position of the log up to which the processing of the compared data is completed. The notation $\Sigma[X; f(x)]$ further means the sum of the values of $f(x)$ over all $x \in X$.

The first condition is a uniform representation of monetary values. For each currency, a positive monetary quantum $\delta(c) > 0$ and a deterministic rounding rule $R(c,\cdot)$ are established.:

$$M(c, x) = Q(c, x) = \delta(c)\, R(c, x / \delta(c)),\ R(c,\cdot): R \to Z. \quad (1)$$

Here, x is the initial result of the calculation. $M(c, x)$ is the monetary value allowed for recording, and $\delta(c)$ is the minimum accounting unit of the selected accuracy. All components must use the same values of $\delta(c)$, $R(c)$, and the moment when rounding is applied. The monetary quantum does not necessarily coincide with the minimum cash face value of the currency. Higher accuracy may be used for internal calculations. Practical implementation involves storing sums in integer minimum units or an exact decimal format (Cowlishaw M. F., Schwarz E. M., Smith R. M., Webb C. F., 2001).

The second condition is that the balance must be balanced for each transaction and currency. For every transaction, the sum of the debits and credits must be equal:

$$\forall o \in O, \forall c \in C: \Sigma[P(o, c); s(p)m(p)] = 0, s(p) \in \{-1, +1\}. \quad (2)$$

Here, $P(o, c)$ is the set of transactions in the currency c, $m(p)$ represents the amount of each transaction in minimum monetary units, and $s(p)$ indicates the side of the record. Different currencies should not offset each other; exchange rate and rounding errors are reflected in separate transactions. Condition (2) is a necessary but not sufficient condition: an erroneous or duplicate transaction can still appear balanced (Ellerman D., 2014).

As a result, for a closed set of accounts N in the same currency c, in the absence of external flows or when the corresponding commission, tax and settlement accounts are included in N, the aggregate invariant $\Sigma[N; b(i, t, c)] = \Sigma[N; b(i, t0, c)]$ holds, where $b(i, t, c)$ – account balance i at time t. This equality is necessary, but not sufficient: mutually compensating errors can save the total amount, therefore, operational control according to formula (2) remains the main one (Ellerman D., 2014).

The third condition is the idempotence of the financial effect. Let s_i^- and s_i^+ denote the states immediately before and after the atomic step of fixing the itch attempt to process the same request with a unique key k and a control value of the normalized content h, and $NPF(S)$ is the financial part of the state. Then

$$|\{i \in \{1,..., n\}: \Pi F(S_i^+) \neq \Pi F(S_i^-)\}| \leq 1. \quad (3)$$

Formula (3) states that any number of attempts to process a request results in no more than one financial change. These repeated attempts may be recorded in the technical journal, as they are not included in the PF forecast. To ensure the practical fulfillment of this condition, it is necessary to atomically store the idempotence key and the financial result. Using the same key with k a different value for h should lead to a conflict and should not be interpreted as a repeat of the original operation. This idempotence guarantees a one-time effect, although it does not guarantee that the request will be processed at least once (Akidau T., Balikov A., Bekiroğlu K., Chernyak S., Haberman J., Lax R., McVeety S., Mills D., Nordstrom P., Whittle S., 2013).

The fourth condition involves serializability in competitive execution with regard to financially observable behavior. $ObsF(H, S(0))$ includes the results of financial transactions, response messages, and the final state of history H. For a history containing n confirmed transactions, there must exist a valid sequence of events π.

$$\exists \pi \in Lin(\prec): ObsF(H, S(0)) = ObsF(Seq(\pi), S(0)). \quad (4)$$

Here, $Seq(\pi)$ represents the sequence of transactions $T(\pi(1)), T(\pi(2)), ..., T(\pi(n))$, executed in order from the initial state $S(0)$. The ratio $\prec$ denotes causal precedence, and $Lin(\prec)$ represents a set of linear continuations of this sequence. $ObsF$ equality ensures that the financial readings, transactions, balances, liabilities, and final status are all consistent. If strict serializability is used, the order in which non-overlapping transactions occur is also taken into account in the $\prec$ relation. This means that an operation that completes before another operation must come before it in the sequence (Papadimitriou C. H., 1979; Lamport L., 1978; Herlihy M. P., Wing J. M., 1990).

For a strict single-object interface, where $\prec$ includes real-time ordering, this requirement corresponds to linearizability. However, its locality does not guarantee the atomicity of transfers associated with multiple accounts. Such an operation should be considered as a single transaction, and not as a set of independent records (Herlihy M. P., Wing J. M., 1990).

In the case of concurrent transactions, the physical order of completion is not necessarily the same as the order in which they were serialized. The global order of all transactions is not always necessary either. An unambiguous order within a single account, contract, or set of transactions that modify a shared financial invariant is sufficient.

The fifth condition is a reconciliation according to the general position of the financial journal. Let $b(J, w)$ be the remainder vector restored from the log to the position w, $b(r, w)$ be the state of the subsystem r, $Proj(r,\cdot)$ be the rule for constructing its representation, $Obj(r, c)$ be the set of objects

to be compared, and $U(J, w)$ and $U(r, w)$ are the sets of identifiers of processed operations in the log and subsystem, respectively. Then

$$D(r, c, w) = \Sigma[Obj(r, c); |Proj(r, b(J, w))(a, c) - b(r, w)(a, c)|] = 0; \quad (5)$$
$$U(J, w) = U(r, w); L(r, w) = t(r, w) - t(J, w) \le \tau(r).$$

The value of $D(r, c, w)$ characterizes the monetary discrepancy. Here, $t(J, w)$ is the time when the position of w was recorded in the log, and $t(r, w)$ is the time the subsystem r reached and provided the state for the same position of w. Therefore, $L(r, w)$ represents the delay in creating the representation, and $\tau(r) > 0$ is its acceptable limit. The use of absolute differences prevents the cancellation of positive and negative errors. Equality of the sets $U(J, w)$ and $U(r, w)$ allows us to identify missing or redundant identifiers. Re-processing is controlled by the uniqueness of the idempotent key.

Comparisons should be made for a single position w. Comparing a current financial journal with a delayed analytical presentation can create a false discrepancy. Formula (5) is the author's reconciliation condition based on the idea of a consistent logical slice of a distributed system (Chandy K. M., Lamport L., 1985).

Conditions (1)–(5) primarily relate to a reliable financial system. In an asynchronous model with the possibility of message loss, it is impossible in all possible scenarios to simultaneously guarantee the linearizability of the read-write object and the execution of each request received by a healthy node (Gilbert, S., Lynch, N., 2002). Thus, in the proposed model, unconfirmed transactions should not be considered committed solely for accessibility reasons. Derived analytical data may still be available and may be temporarily delayed, but they should not be used to confirm financial performance. Their status should be accepted only after agreement in accordance with the common position.

Thus, formulas (1)–(4) define the invariants of financial transaction security, and formula (5) is a verifiable condition for the convergence of derived representations. Financial accuracy in this model does not imply the continuous physical identity of all records, but rather the preservation of a reproducible history, causal relationships, and controlled convergence of data. None of these conditions alone proves the validity of an economic transaction, but their combined implementation allows for the detection of internal contradictions, duplications, disruptions, and incomplete distributions of operations.

The main stages of financial transaction processing are outlined in Table 3.

Table 3. *The main stages of financial transaction processing*

Stage	Content	Result
Accepting the request	Receiving and parsing input data	Generated set of banking details
Checking banking details	Control of required fields and formats	Admission to further processing or refusal
Checking conditions	Control of authority, restrictions, and data availability	Confirmation of the possibility of execution
Transaction execution	Modification of related financial records	Preliminary result of the operation
Completion	Confirming or canceling changes	Fixed or restored state
Status formation	Specifying the result and the reason for the failure if there is an error	Operation status message
Execution control	Checking the completion of processing in all involved components	Confirmation of the final status

A source: author's development based on (Gorbunova A. V., 2025; Tatarnikova T. M., Shelest M. N., 2022; Fomin D. S., Balsamov A. V., 2021)

When implementing the model, it is important to establish the mandatory details of each operation in advance, as well as the rules for verifying them and the acceptable processing states. Verification results should be clear and unambiguous, with reasons for rejection available for analysis. For complex local operations, it may be advisable to use save points that allow for cancellation of only the erroneous part of a transaction. It is also important to determine a procedure in case of component failure, as well as establish recovery indicators.

The recovery time determines the maximum acceptable period of unavailability for a resource, while the backup frequency should consider the goals for recovery time and acceptable data loss. These values are set by the organization based on an analysis of the impact of failures on their activities.

Special attention should be given to testing after making changes to the software, data structure, or the order of processing operations. The Basel Committee emphasizes that issues related to quality and data aggregation must be taken into consideration when developing new products and making modifications to information systems. Additionally, the architecture should ensure the ability to process and prepare data even under conditions of increased workload or crisis situations (Basel Committee on Banking Supervision. 2013).

Continuous monitoring should cover all aspects: computing power, software, work environment, data, unauthorized access attempts, and actions by external service providers. Any detected abnormalities should be analyzed at an early stage, as early detection minimizes possible consequences and reduces the time required for recovery.

Conclusions

The research has shown that achieving financial accuracy in a distributed system requires the coordinated effort of all components involved in the receipt, processing, registration, and transmission of financial data. Our proposed model incorporates transactional processing, event logging in sequence, message transmission, control of execution, and recovery from crashes.

By utilizing unique identifiers, idempotent processing, recording of versions, and reconciliation of records, we can prevent duplicate changes to financial results, detect conflicting transactions, and identify the origins of discrepancies. To ensure the practical implementation of this model, it is essential to establish rules for verifying bank details, monitor the status of operations, test the system after modifications, and continuously monitor its performance. This approach helps reduce the risk of errors and ensures the reliability of the financial condition in the distributed system.

References

Gorbunova A. V. (2025). Evaluation of the model of distributed transactional applications with microservice architecture and connections // Information Management Systems. 2025. – No. 6 (139). – P. 42–50. DOI 10.31799/1684-8853-2025-6-42-50.

Kravchenko D. A. (2022). Microservice architecture // Interactive Science. 2022. – No. 4 (69). – P. 43–44. DOI 10.21661 / p-556565.

Kuznetsov S. D., Poskonin A. V. (2013). Distributed horizontally scalable solutions for data management // Transactions of the Institute for System Programming of the Russian Academy of Sciences. 2013. – Vol. 24. – P. 327–358.

Malyuga K. V., Perl I. A., Slapoguzov A. P. (2021). Evaluation of the applicability of asynchronous programming methods in ensuring data consistency problems in a microservice environment // Scientific and Technical Bulletin of Information Technologies, Mechanics and Optics. 2021. – Vol. 21. – No. 4. – P. 473–481. DOI 10.17586/2226-1494-2021-21-4-473-481.

Popov S. G., Fridman V. S. (2018). Review of methods of dynamic data distribution in distributed database management components // Scientific and Technical Bulletin of St. Petersburg State Polytechnical University. Computer Science. Telecommunications. Management. 2018. – Vol. 11. – No. 4. – P. 82–107. DOI 10.18721/JCSTCS.11407.

Soloviev A. V. (2025). Organization of data storage in distributed DBMS with strict data consistency // Information technologies and computing systems. 2025. – No. 2. – P. 113–122. DOI 10.14357/20718632250210.

Sukhoplyuev D. I., Nazarov A. N. (2024). Monitoring fault tolerance in distributed networks // Computational nanotechnologies. 2024. – Vol. 11. – No. 4. – P. 94–106. DOI 10.33693/2313–223X-2024-11-4-94-106.

Tatarnikova T. M. (2023). On-the-fly data clustering for PostgreSQL DBMS // Software products and systems. 2023. – No. 2. – P. 196–201. DOI 10.15827/0236-235X.142.196-201.

Tatarnikova T. M., Shelest M. N. (2022). Evaluation of boundary results of the average response time to a user information system request // Software products and systems. 2022. – No. 3. – P. 488–492. DOI 10.15827/0236–235X.139.488-492.

Fomin D. S., Balsamov A. V. (2021). "The Problem of Transaction Processing Using a Microservice Design" // News of Higher Educational Institutions. Volga Region. Engineering Sciences. 2021. – No. 2 (58). – P. 15–23. DOI 10.21685/2072-3059-2021-2-2.

Cowlishaw M. F., Schwarz E. M., Smith R. M., Webb C. F. (2001). "Specification of Decimal Floating-Point Numbers" // Proceedings of the 15th IEEE Symposium on Computer Arithmetic. 2001. – P. 147–154. DOI 10.1109/ARITH.2001.930114.

Ellerman D. (2014). On Double-Entry Accounting: A Mathematical Approach // Accounting Education. 2014. – Vol. 23. – No. 5. – P. 483–501. DOI 10.1080/09639284.2014.949803.

Akidau T., Balikov A., Bekiroğlu K., Chernyak S., Haberman J., Lax R., McVeety S., Mills D., Nordstrom P., Whittle S. (2013). Mill Wheel: Fault-Tolerant Stream Processing at Internet Scale // Transactions of the VLDB Foundation. 2013. – Vol. 6. – No. 11. – P. 1033–1044. DOI 10.14778/2536222.2536229.

Papadimitriou C. H. (1979). Serializability of Concurrent Database Updates // Journal of the ACM. 1979. – Vol. 26. – No. 4. – P. 631–653. DOI 10.1145/322154.322158.

Lamport L. (1978). Time, Clocks, and Event Ordering in a Distributed System // Communications of the ACM. 1978. – Vol. 21. – No. 7. – P. 558–565. DOI 10.1145/359545.359563.

Herlihy M. P., Wing J. M. (1990). Linearizability: A Correctness Condition for Parallel Objects // ACM Transactions on Programming Languages and Systems. – 1990. – Vol. 12, No. 3. – pp. 463–492. – DOI 10.1145/78969.78972.

Chandy K. M., Lamport L. (1985). Distributed Snapshots: Determining Global States of Distributed Systems // ACM Transactions on Computer Systems. 1985. – Vol. 3. – No. 1. – P. 63–75. DOI 10.1145/214451.214456.

Bailis P., Fekete A., Franklin M. J., Ghodsi A., Hellerstein J. M., Stoica I. (2014). Coordination Avoidance in Database Systems // Proceedings of the VLDB Foundation. 2014. – Vol. 8. No. 3. – P. 185–196. DOI 10.14778/2735508.2735509.

Bação J. L., Guerreiro S. (2026). Consistency challenges in event-driven microservices: A literature review on the evolution of data, processes, and systems // Computing engineering. 2026. – Vol. 108. – Article 110. DOI 10.1007/s00607-026-01701-5.

Basel Committee on Banking Supervision. Principles for Effective Risk Data Aggregation and Reporting. – Basel: Bank for International Settlements, 2013. URL: https://www.bis.org/publ/bcbs239.pdf.

Amazon Web Services. "Transactional Outbox Pattern" // AWS Recommendations. –URL: https://docs.aws.amazon.com/prescriptive-guidance/latest/cloud-design-patterns/transactional-outbox.html.

Gilbert, S., Lynch, N. (2002). "Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services" // ACM SIGACT News. 2002. – Vol. 33. – No. 2. – P. 51–59. DOI 10.1145/564585.564601.

submitted 17.07.2026;
accepted for publication 31.07.2026;
published 31.07.2026

Contact: impxstudio@gmail.com