УДК 004.65:004.738.5:339.371
DOI https://doi.org/10.32782/tnv-tech.2025.6.13

БАЗИ ДАНИХ У ВЕБСЕРВІСАХ ЕЛЕКТРОННОЇ КОМЕРЦІЇ: АРХІТЕКТУРНІ РІШЕННЯ ТА ПРОБЛЕМИ МАСШТАБОВАНОСТІ

Осадчук С. І. *– викладач кафедри комп'ютерних систем і технологій*
Приватного вищого навчального закладу «Буковинський університет»
ORCID ID: 0009-0000-0203-2581

Федорчук А. Л. *– кандидат педагогічних наук, доцент,*
доцент кафедри комп'ютерних наук та інформаційних технологій
Житомирського державного університету імені Івана Франка
ORCID ID: 0000-0001-8227-3210

Батаєв С. В. *– аспірант Ворицького університету*
ORCID ID: 0009-0009-9564-9403

Здійснено дослідження архітектурних підходів баз даних типів SQL, NoSQL та NewSQL для функціональних модулів платформ електронної комерції, вивчено технології реплікації інформації, розподілу даних через ключі партіціювання та розподілу навантаження з метою реалізації горизонтального масштабування у системах з пропускною здатністю понад 500,000 запитів на секунду.

Мета статті – вивчити сучасні архітектурні підходи до організації баз даних у веб-сервісах електронної комерції та дослідити технічні виклики масштабованості систем керування даними за умов зростаючого навантаження.

Наукова новизна полягає у всебічному дослідженні кореляції між обранням типу системи управління базами даних (реляційні СУБД із ACID-властивостями, документоорієнтовані NoSQL із гнучкою схемою, key-value сховища для операцій у пам'яті) та можливостями горизонтального масштабування для функціональних компонентів систем електронної комерції (каталог продуктів із переважанням операцій читання, процесинг замовлень із транзакційною узгодженістю, кешування користувацьких сесій). Проаналізовано архітектурний патерн Database per Service у контексті мікросервісної архітектури та проблематику розподілених транзакцій через застосування Saga-патерну.

Результати демонструють, що гібридні підходи з використанням MERN-стеку (MongoDB для каталогів з 100K-500K reads/sec per shard), CockroachDB для геолокаційно розподілених транзакцій з automatic sharding по 64 MB, Redis Cluster для кешування з throughput 1-10 мільйонів ops/sec забезпечують оптимальний баланс між продуктивністю та масштабованістю. Доведено, що впровадження Master-Slave реплікації підвищує доступність до 99.95% та знижує час відповіді на 65-72% при 10,000+ одночасних користувачів. Архітектура з 10-15 read replicas дозволяє масштабувати читання лінійно з aggregate throughput 500K-1M reads/sec. Геолокаційне партиціонування знижує латентність до 10-50 мс замість 150-300 мс cross-continental. Database Reliability Engineering практики скорочують MTTR на 80%.

Висновки. Для масштабованості баз даних вебсервісів електронної комерції необхідно використовувати комбінацію реляційних СУБД для ACID-транзакцій, NoSQL-систем для високопродуктивного читання та in-memory баз для кешування, з мікросервісною архітектурою, автоматичною реплікацією, горизонтальним шардингом та багаторівневим кешуванням з cache hit ratio 85-90%.

Ключові слова: *масштабованість баз даних, мікросервісна архітектура, e-commerce платформи, NoSQL системи, розподілені бази даних.*

Osadchuk S. I., Fedorchuk A. L., Bataiev S. V. Databases in e-commerce web services: architectural solutions and scalability issues

The architectural patterns of SQL, NoSQL, and NewSQL databases for various components of e-commerce platforms are analyzed, and the mechanisms of data replication, sharding via partition keys, and load balancing are investigated to ensure horizontal scaling of systems with a throughput of over 500,000 requests per second.

The purpose of the article is to investigate modern database architectural solutions in e-commerce web services and analyze the technical problems of scalability of data management systems under increasing loads.

The scientific novelty lies in a comprehensive analysis of the relationship between the choice of database type (relational DBMS with ACID guarantees, document-oriented NoSQL with flexible schema, key-value stores for in-memory operations) and the scalability capabilities of functional modules of e-commerce systems (product catalog with read-heavy workload, order processing with transactional consistency, session caching). We investigated the Database per Service pattern in microservice architecture and the problem of distributed transactions through the Saga pattern.

The results demonstrate that hybrid approaches using the MERN stack (MongoDB for catalogs with 100K-500K reads/sec per shard), CockroachDB for geolocation-distributed transactions with automatic sharding at 64 MB, and Redis Cluster for caching with a throughput of 1-10 million ops/sec provide an optimal balance between performance and scalability. It has been proven that the implementation of Master-Slave replication increases availability to 99.95% and reduces response time by 65-72% with 10,000+ concurrent users. An architecture with 10-15 read replicas allows for linear scaling of reads with an aggregate throughput of 500K-1M reads/sec. Geolocation partitioning reduces latency to 10-50 ms instead of 150-300 ms cross-continental. Database Reliability Engineering practices reduce MTTR by 80%.

Conclusions. For the scalability of e-commerce web service databases, it is necessary to use a combination of relational DBMS for ACID transactions, NoSQL systems for high-performance reading, and in-memory databases for caching, with microservice architecture, automatic replication, horizontal sharding, and multi-level caching with a cache hit ratio of 85-90%.

Key words: *Database scalability, microservice architecture, e-commerce platforms, NoSQL systems, distributed databases.*

Вступ. Платформи електронної комерції працюють в умовах експоненційного приросту інформаційних обсягів, що потребує інноваційних архітектурних стратегій побудови систем управління базами даних. Класичні монолітні архітектури неспроможні забезпечити потрібну пропускну здатність понад 500,000 запитів на секунду в періоди пікового навантаження. Впровадження розподілених систем баз даних стало технологічною необхідністю для досягнення масштабованості.

Архітектурні концепції мають враховувати високий рівень доступності (99.99% uptime), горизонтальне масштабування через розподіл даних із ключами партіціювання, розподіл навантаження засобами HAProxy або ProxySQL, латентність на рівні sub-millisecond для операцій у пам'яті та географічно розподілене розміщення даних. Виклики масштабованості систем управління базами даних набувають критичного значення при досягненні пропускної здатності сотень тисяч транзакцій за секунду.

Постановка проблеми. Основною проблемою є вибір архітектури бази даних для різних компонентів e-commerce систем. Реляційні бази (PostgreSQL, MySQL) забезпечують ACID-гарантії, однак погано масштабуються горизонтально. NoSQL рішення (MongoDB, Cassandra, Redis) демонструють горизонтальну масштабованість через partition keys та consistent hashing з продуктивністю 10,000-100,000 ops/sec, але працюють за eventual consistency моделлю.

Пікові навантаження створюють до 500,000-1,000,000 запитів за секунду. Вертикальне масштабування має обмеження (128 ядер, 2 TB RAM), тоді як горизонтальне через шардинг вимагає спеціальної архітектури. Каталог товарів демонструє read-heavy workload (read:write 95:5) та потребує кешування з cache hit ratio 85-90% для sub-millisecond латентності. Система замовлень потребує

ACID-гарантій для транзакційної консистентності. Перехід до мікросервісної архітектури з Database per Service вимагає вирішення проблеми розподілених транзакцій.

Аналіз останніх досліджень і публікацій. Petrukha N., Zhmaiev A., Synkevych M. [1, с. 830] підкреслюють важливість ітеративного підходу до проектування масштабованих баз даних з можливістю адаптації схеми без простою системи. Akula A. K. [2, p. 1055] демонструє, як хмарні рішення (RDS Multi-AZ, DynamoDB з partition keys, ElastiCache для sub-millisecond кешування) дозволяють обробляти 300+ мільйонів користувачів з масштабуванням від 2 до 50+ вузлів та throughput 500,000+ запитів за секунду. Ileana M., Oproiu M. I., Marian C. V. [3, p. 996] підкреслюють переваги Event Sourcing та CQRS для розділення read-heavy workload (95% reads) від write-intensive операцій. Shwetha H., Prajwal D., Sridharan S. [4, p. 3] демонструють MongoDB з automatic sharding по partition keys для досягнення 100K-500K reads/sec per shard. Ratra K. K., Kumar Seth D., Uppuluri S. [5, p. 4] аналізують оптимізацію через автоматичне масштабування при SLA 99.9% та обробці 500,000+ concurrent connections.

Taft R., Sharif I., Matei A., VanBenschoten N., Lewis J., Grieger T., Niemi K., Woods A., Birzin A., Poss R., Bardea P., Ranade A., Darnell B., Gruneir B., Jaffray J., Zhang L., Mattis P. [6, p. 1495] представляють CockroachDB з range-based sharding (ranges по 64 MB) та multi-raft consensus, забезпечуючи throughput до 400K transactions/sec при латентності 10-50 мс. Shekhar Mishra [7, p. 1325] підкреслює автоматизацію через MHA або Orchestrator з RTO під 30 секунд для підтримки throughput 500,000+ запитів. Bhola M., Bajeja S. [8, p. 3] демонструють read replicas до 15 replicas для linear scaling read-heavy workload та provisioned IOPS до 80,000.

Pratama B. P. A., Gunawan Hardianto B., Dharmayanti D. [9, p. 2230] досліджують балансування через HAProxy та ProxySQL для MySQL з Master-Slave реплікацією (binlog lag 1-3 секунди), автоматичний failover через MHA або Orchestrator з downtime 20-30 секунд, досягаючи aggregate throughput 180K queries/sec та availability 99.95%. Anusha Reddy Guntakandla [10, p. 645] аналізує Database per Service для незалежного масштабування та fault isolation.

Виклад основного матеріалу. Для досягнення throughput понад 500,000 запитів за секунду використовують polyglot persistence – комбінацію SQL (MySQL 8.0, PostgreSQL 15) для ACID-транзакцій, NoSQL (MongoDB 7.0 з sharding, Cassandra 4.0 з consistent hashing, Redis 7.2 для sub-millisecond кешування) для read-heavy workload, та NewSQL (CockroachDB 23.0) для geo-distributed consistency. Akula A. K. [2, p. 1058] демонструє архітектурний підхід: DynamoDB з partition keys (product_id як hash key) для каталогу забезпечує латентність 1-10 мс при 10 мільйонах reads/sec через auto-sharding, RDS PostgreSQL для транзакцій з SERIALIZABLE isolation, ElastiCache Redis з sub-millisecond латентністю 0.5-2 мс та cache hit ratio 85-90%, throughput до 25 мільйонів ops/sec.

Мікросервісна архітектура з Database per Service дозволяє оптимізувати кожен компонент. Anusha Reddy Guntakandla [10, p. 646] підкреслює: Product Catalog Service з MongoDB використовує hash-based sharding через partition key product_id для 100K-500K reads/sec per shard в read-heavy workload (95% reads), Order Management Service з PostgreSQL забезпечує ACID для транзакційних workflows, Inventory Service з CockroachDB та range-based partition keys забезпечує strong consistency для real-time stock updates. Для систематизації розглянемо таблицю 1, що порівнює різні типи баз даних за параметрами масштабованості.

Таблиця 1

Порівняльна характеристика типів баз даних для e-commerce платформ

Тип БД	Механізм масштабування	Partition strategy	Throughput читання	Latency
SQL (MySQL, PostgreSQL)	Вертикальне (до 128 cores, 2TB RAM)	N/A (monolithic)	50K-200K ops/sec	5-50 мс
NoSQL Document (MongoDB)	Горизонтальне через partition keys	Hash/Range sharding	100K-500K per shard	1-10 мс
NoSQL Key-Value (Redis)	Consistent hashing (1000 nodes)	Hash slots (16384)	1M-10M per node	0.1-2 мс (sub-ms)
NoSQL Column (Cassandra)	Consistent hashing (1000+ nodes)	Partition key + clustering	200K-1M per node	2-20 мс
NewSQL (CockroachDB)	Range-based partition keys (64MB)	Automatic range splits	100K-400K per cluster	5-30 мс

Аналізуючи таблицю 1, для досягнення throughput понад 500,000 запитів необхідна комбінація різних типів баз даних. NoSQL document stores з hash-based partition keys оптимальні для каталогів з read-heavy workload та throughput 100K-500K reads/sec per shard. Redis з consistent hashing через 16384 hash slots забезпечує sub-millisecond латентність 0.1-2 мс та cache hit ratio 85-90%, обробляючи 70-80% запитів з кешу без звернення до primary database. Shwetha H., Prajwal D., Sridharan S. [4, p. 4] демонструють MongoDB з sharding по partition key category_id або product_id, де hash-based sharding використовує формулу: shard = hash(partition_key) mod num_shards, забезпечуючи рівномірний розподіл даних та throughput 100K-500K reads/sec per shard.

Масштабованість досягається через комбінацію шардингу, реплікації та балансування. Ratra K. K., Kumar Seth D., Uppuluri S. [5, p. 5] аналізують, що вертикальне масштабування має hard limits (128 cores, 2TB RAM, 80K IOPS) та не забезпечує throughput понад 200K queries/sec. Горизонтальне масштабування через sharding з partition keys забезпечує linear scaling: 10 shards × 50K queries/sec = 500K aggregate throughput. Реплікація забезпечує високу доступність та розподілення read-heavy workload. Bhola M., Bajeja S. [8, p. 6] описують AWS RDS Multi-AZ з synchronous replication між primary та standby для zero data loss (RPO=0) та automated failover з RTO 60-120 секунд, плюс asynchronous read replicas (до 15 replicas) для horizontal scaling – кожна replica додає 50K-200K reads/sec, забезпечуючи aggregate throughput 500K-1M reads/sec.

Pratama B. P. A., Gunawan Hardianto B., Dharmayanti D. [9, p. 2234] демонструють практичну реалізацію з HAProxy для MySQL: конфігурація 1 master + 10 read replicas збільшила aggregate throughput з 45K до 520K queries/sec, знизила average latency з 850 мс до 240 мс при 10,000 одночасних користувачів, та забезпечила automated failover через MHA з downtime 25 секунд, підвищуючи availability до 99.95%. Taft R. та співавтори [6, p. 1500] пояснюють CockroachDB range-based sharding: кожна table автоматично розбивається на ranges по 64 MB, ranges автоматично split при досягненні size threshold та distribute між nodes. Кожен range реплікується на 3-5 nodes через Raft consensus, забезпечуючи linearizable consistency. Cluster з 50 nodes досягає 400K aggregate throughput.

Перехід до мікросервісної архітектури створює проблему distributed transactions. Ileana M., Oproiu M. I., Marian C. V. [3, p. 998] підкреслюють, що Two-Phase Commit protocol не scale понад 10K-20K transactions/sec через blocking

nature та increased latency (додає 2 network round-trips). Saga-патерн вирішує проблему через sequence локальних транзакцій з compensating actions. Anusha Reddy Guntakandla [10, p. 647] описує order checkout saga: Order Service створює order (10-20 мс), Payment Service charges (100-500 мс), Inventory Service decrements stock (10-20 мс), Shipping Service створює label (200-800 мс). Total latency 370-1440 мс vs 50-100 мс для monolithic ACID transaction, але дозволяє independent scaling кожного service. Таблиця 2 систематизує trade-offs між архітектурами.

Таблиця 2

Порівняння монолітної та мікросервісної архітектури баз даних

Характеристика	**Монолітна архітектура**	**Мікросервісна архітектура (Database per Service)**
Масштабування	Вертикальне (до 128 cores, 2TB RAM)	Горизонтальне через independent service scaling
Throughput ceiling	50K-200K queries/sec (single instance)	500K-1M+ aggregate через N services × throughput
Transaction model	ACID з 2PC protocol (50-100 мс)	Saga pattern з eventual consistency (370-1440 мс)
Consistency	Strong consistency (linearizable)	Eventual consistency (100-500 мс lag)
Fault isolation	Single point of failure (SPOF)	Isolated failures не впливають на інші services
Operational complexity	Single database management	N databases + service mesh overhead

Архітектура мікросервісів забезпечує сукупну пропускну здатність понад 500,000 запитів завдяки можливості незалежного масштабування компонентів. Модуль Product Catalog досягає 300K операцій читання за секунду на базі MongoDB. Сервіс Order Service обробляє 50K транзакцій за секунду з використанням PostgreSQL. Компонент Inventory виконує 100K оновлень за секунду на платформі CockroachDB. Застосування Saga-патерну призводить до збільшення затримки в діапазоні від 370 до 1440 мілісекунд. Така латентність вимагає впровадження асинхронної обробки операцій.

Akula A. K. [2, p. 1059] демонструє multi-tier caching: browser cache (TTL 1-7 днів), CDN (TTL 1-24 години, latency 10-30 мс), Redis (TTL 5-60 хвилин, latency 0.5-2 мс). Redis Cluster з 16384 hash slots забезпечує 1-10 мільйонів ops/sec. Cache hit ratio 85-90% знижує database load на 80-90%.

Taft R. та співавтори [6, p. 1502] демонструють CockroachDB geo-partitioning з латентністю 10-50 мс intra-region проти 150-300 мс cross-continental. Synchronous replication через Raft додає commit latency 50-150 мс, asynchronous знижує до 10-30 мс з eventual consistency window 100-500 мс.

Shekhar Mishra [7, p. 1326] систематизує DBRE: continuous monitoring (query latency p99, replication lag, CPU/memory), automated failover через MHA (detection 10 сек + promotion 15-20 сек = RTO < 30 сек), chaos engineering (kill nodes, inject latency 100-500 мс, simulate disk full). Chaos testing скорочує MTTR на 80%.

Висновки. Для throughput понад 500,000 запитів за секунду необхідна polyglot persistence: PostgreSQL/MySQL для ACID-транзакцій, MongoDB з hash-based sharding (100K-500K reads/sec per shard), Redis Cluster з 16384 hash slots (cache hit

ratio 85-90%, латентність 0.5-2 мс). Master-Slave реплікація з 10-15 read replicas забезпечує aggregate throughput 500K-1M reads/sec. HAProxy/ProxySQL з automated failover (RTO < 30 секунд) підвищує availability до 99.95%.

Мікросервісна архітектура з Database per Service забезпечує independent scaling. Saga-патерн збільшує latency до 370-1440 мс проти 50-100 мс monolithic ACID. CockroachDB з range-based sharding по 64 MB досягає 100K-400K transactions/sec з linearizable consistency. Геолокаційне партиціонування знижує latency до 10-50 мс intra-region проти 150-300 мс cross-continental. DBRE практики скорочують MTTR на 80%.

Дослідження serverless databases (Aurora Serverless v2) для automatic scaling з millisecond granularity. Аналіз vector databases (Pinecone, Milvus) для AI-driven recommendations з 10K-100K vector queries/sec. Оптимізація distributed SQL (Spanner TrueTime, YugabyteDB) для global consistency з latency < 10 мс. Energy efficiency через workload-aware resource allocation для зниження operational costs на 30-50%. Multi-cloud database federation для disaster recovery через cross-cloud replication.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

1. Petrukha N. Innovative Approaches to it Project Management Using Agile Project and Management Methods. *Science and Technology Today*. 2024. № 8(36). P. 824-839. DOI: https://doi.org/10.52058/2786-6025-2024-8(36)-824-839.

2. Akula A. K. Leveraging AWS and Java Microservices: An Analysis of Amazon's Scalable E-commerce Architecture. *International Journal for Research in Applied Science and Engineering Technology*. 2024. Vol. 12, № 9. P. 1051-1063. DOI: https://doi.org/10.22214/ijraset.2024.64275. URL: https://www.ijraset.com/best-journal/leveraging-aws-and-java-microservices-an-analysis-of-amazons-scalable-ecommerce-architecture.

3. Ileana M. E-commerce Solutions using Distributed Web Systems with Microservices-Based Architecture for High-Performance Online Stores. 2024 47th MIPRO ICT and Electronics Convention (MIPRO). 2024. P. 994–999. DOI: https://doi.org/10.1109/mipro60963.2024.10569273. URL: https://ieeexplore.ieee.org/document/10569273.

4. Shwetha H. From MongoDB to React: Unleashing the Power of MERN in E-commerce. 2024 International Conference on Emerging Technologies in Computer Science for Interdisciplinary Applications (ICETCS). 2024. P. 1-6. DOI: https://doi.org/10.1109/icetcs61022.2024.10543521. URL: https://ieeexplore.ieee.org/document/10543521.

5. Ratra K. K. Energy-Efficient Microservices Architecture for Large-Scale E-Commerce Platforms. 2025 IEEE Conference on Technologies for Sustainability (SusTech). 2025. P. 1-8. DOI: https://doi.org/10.1109/sustech63138.2025.11025688. URL: https://ieeexplore.ieee.org/document/11025688.

6. Taft R. CockroachDB: The Resilient Geo-Distributed SQL Database. Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020. P. 1493-1509. DOI: https://doi.org/10.1145/3318464.3386134. URL: https://dl.acm.org/doi/10.1145/3318464.3386134.

7. Shekhar Mishra. Building Scalable Cloud Databases with Database Reliability Engineering. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 2025. Vol. 11, № 1. P. 1322–1333. DOI: https://doi.org/10.32628/cseit251112125. URL: https://ijsrcseit.com/index.php/home/article/view/CSEIT251112125.

8. Bhola M. Enhancing Cloud-Native Relational Database Systems: Proposed Design Patterns for AWS RDS Application. *SN Computer Science*. 2025. Vol. 6,

№ 5. DOI: https://doi.org/10.1007/s42979-025-04055-5. URL: https://link.springer.com/article/10.1007/s42979-025-04055-5.

9. Pratama B. P. A. Implementation of High Availability Based on Load Balancing and Failover Method in E-commerce Using MySQL Database. *Asian Journal of Engineering, Social and Health.* 2024. Vol. 3, № 10. P. 2226–2239. DOI: https://doi.org/10.46799/ajesh.v3i10.419. URL: https://ajesh.ph/index.php/gp/article/view/419.

10. Anusha Reddy Guntakandla. Microservices Architecture: Decomposing E-Commerce Monoliths into Scalable, Independent Services. *Journal of Information Systems Engineering and Management*. 2025. Vol. 10, № 58s. P. 642-650. DOI: https://doi.org/10.52783/jisem.v10i58s.12665. URL: https://www.jisem-journal.com/index.php/journal/article/view/12665.

REFERENCES:

1. Petrukha N., Zhmaiev A., Synkevych M. Innovative Approaches to it Project Management Using Agile Project and Management Methods. Science and Technology Today. 2024. № 8(36). P. 824–839. DOI: https://doi.org/10.52058/2786-6025-2024-8(36)-824-839 [in English].

2. Akula, A. K. (2024). Leveraging AWS and Java Microservices: An Analysis of Amazon's Scalable E-commerce Architecture. *International Journal for Research in Applied Science and Engineering Technology*, 12(9), 1051–1063. DOI: https://doi.org/10.22214/ijraset.2024.64275 Retrieved from https://www.ijraset.com/best-journal/leveraging-aws-and-java-microservices-an-analysis-of-amazons-scalable-ecommerce-architecture [in English].

3. Ileana, M., Oproiu, M. I., & Marian, C. V. (2024). E-commerce Solutions using Distributed Web Systems with Microservices-Based Architecture for High-Performance Online Stores. 2024 47th MIPRO ICT and Electronics Convention (MIPRO), 994–999. DOI: https://doi.org/10.1109/mipro60963.2024.10569273 Retrieved from https://ieeexplore.ieee.org/document/10569273 [in English].

4. Shwetha, H., Prajwal, D., & Sridharan, S. (2024). From MongoDB to React: Unleashing the Power of MERN in E-commerce. 2024 International Conference on Emerging Technologies in Computer Science for Interdisciplinary Applications (ICETCS), 1–6. DOI: https://doi.org/10.1109/icetcs61022.2024.10543521 Retrieved from https://ieeexplore.ieee.org/document/10543521 [in English].

5. Ratra, K. K., Kumar Seth, D., & Uppuluri, S. (2025). Energy-Efficient Microservices Architecture for Large-Scale E-Commerce Platforms. 2025 IEEE Conference on Technologies for Sustainability (SusTech), 1–8. DOI: https://doi.org/10.1109/sustech63138.2025.11025688 Retrieved from https://ieeexplore.ieee.org/document/11025688 [in English].

6. Taft, R., Sharif, I., Matei, A., VanBenschoten, N., Lewis, J., Grieger, T., Niemi, K., Woods, A., Birzin, A., Poss, R., Bardea, P., Ranade, A., Darnell, B., Gruneir, B., Jaffray, J., Zhang, L., & Mattis, P. (2020). CockroachDB: The Resilient Geo-Distributed SQL Database. Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 1493–1509. DOI: https://doi.org/10.1145/3318464.3386134 Retrieved from https://dl.acm.org/doi/10.1145/3318464.3386134 [in English].

7. Shekhar Mishra. (2025). Building Scalable Cloud Databases with Database Reliability Engineering. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 11(1), 1322–1333. DOI: https://doi.org/10.32628/cseit251112125 Retrieved from https://ijsrcseit.com/index.php/home/article/view/CSEIT251112125 [in English].

8. Bhola, M., & Bajeja, S. (2025). Enhancing Cloud-Native Relational Database Systems: Proposed Design Patterns for AWS RDS Application. SN Computer Science, 6(5). DOI: https://doi.org/10.1007/s42979-025-04055-5 Retrieved from https://link.springer.com/article/10.1007/s42979-025-04055-5 [in English].

9. Pratama, B. P. A., Gunawan Hardianto, B., & Dharmayanti, D. (2024). Implementation of High Availability Based on Load Balancing and Failover Method in E-commerce Using MySQL Database. *Asian Journal of Engineering*, Social and Health, 3(10), 2226–2239. DOI: https://doi.org/10.46799/ajesh.v3i10.419 Retrieved from https://ajesh.ph/index.php/gp/article/view/419 [in English].

10. Anusha Reddy Guntakandla. (2025). Microservices Architecture: Decomposing E-Commerce Monoliths into Scalable, Independent Services. *Journal of Information Systems Engineering and Management, 1*0(58s), 642–650. DOI: https://doi.org/10.52783/jisem.v10i58s.12665 Retrieved from https://www.jisem-journal.com/index.php/journal/article/view/12665 [in English].

Дата першого надходження рукопису до видання: 20.11.2025
Дата прийнятого до друку рукопису після рецензування: 15.12.2025
Дата публікації: 30.12.2025