Open Journal of Physical Science (OJPS)
ISSN: 2734-2123
Article Details:
DOI: 10.52417/ojps.v6i2.1050
Article Ref. No.: OJPS0602006-918
Volume: 6; Issue: 2, Pages: 79 - 96 (2025)
Accepted Date: 25th September, 2025

Open Journals Nigeria (OJN)
Open Access | Bi-annual | Peer-reviewed
www.openjournalsnigeria.org.ng
editorial@openjournalsnigeria.org.ng

REVIEW ARTICLE **OJPS0602006-918**

A SYSTEMATIC REVIEW OF CONCURRENCY CONTROL AND RECOVERY MECHANISMS IN DISTRIBUTED DATABASE SYSTEMS: TRENDS, TECHNIQUES, AND PERFORMANCE MODELLING.

[1]Okoronkwo, M., [1]Chibunze, N., [*2]Markus, C., [1]Adaji, M. A. & [1]Mustapha, M.

[1]Department of Computer Science, University of Nigeria, Nsukka, Nigeria
[2]Department of Computer Science, Taraba State University, Jalingo, Nigeria

***Corresponding Author Email:** calebmarkus@tsuniversity.edu.ng*

ABSTRACT

Concurrency control and recovery mechanisms are essential for ensuring reliability and consistency in distributed database systems (DDBS). This systematic review examines recent advances between 2020 and 2025, focusing on trends, techniques, and performance models. Following the PRISMA protocol, studies were sourced from the Association for Computing Machinery (ACM) Digital Library, the Institute of Electrical and Electronics Engineers (IEEE) Xplore, SpringerLink, and ScienceDirect using targeted keywords and strict inclusion criteria. Concurrency control approaches are grouped into pessimistic methods, such as Two-Phase Locking (2PL) with high latency under contention, optimistic methods offering higher throughput but increased abort rates, multiversion systems providing read scalability, and adaptive machine learning hybrids that reduce latency in conflict conditions. Recovery mechanisms include Write Ahead Logging (WAL) with minimal performance loss, checkpointing and shadow paging with moderate degradation, and distributed commit protocols like the Paxos consensus protocol that maintain high availability. Performance modelling through queuing theory and Markov chains shows that Optimistic Concurrency Control (OCC) has higher rollback probabilities under heavy load, while Two Phase Locking (2PL) reaches throughput limits at high transaction rates. The review offers a decision framework linking workload characteristics to suitable mechanisms and highlights gaps such as inconsistent benchmarking and limited industrial validation. Future research should focus on unified benchmarking, intelligent adaptive models, and resilient recovery methods for distributed environments.

Keywords: *Distributed Database Systems, Concurrency Control, Recovery Mechanisms, Performance Modelling, Fault Tolerance, Systematic Review.*

INTRODUCTION

Distributed Database Systems (DDBS) form the backbone of modern data-intensive applications, spanning cloud native platforms, blockchain-based infrastructures, and large-scale service orchestration for vertical industries such as 5G and energy grids (Zeydan *et al.,* 2022; Zang *et al.,* 2024; Roozbehani *et al.,* 2022). By design, DDBS distributes storage and processing across multiple nodes, often in heterogeneous and geographically dispersed environments, to enhance scalability, fault tolerance, and load balancing (Miryala, 2024; Vatjalainen, 2023). However, the decentralised nature of these systems introduces critical challenges in concurrency control, ensuring simultaneous transactions execute without violating ACID principles, and in recovery mechanisms that maintain operational continuity following node, network, or process failures (Oloruntoba, 2025; Mendonça *et al.,* 2020). The rapid rise of edge computing and multicloud orchestration has further heightened the demand for systems that can sustain high availability (99.99% uptime), strong consistency, and sub-second recovery times under extreme load conditions (Zang and Kim, 2023; Girhotra and Byrisetty, 2025). This review positions concurrency control and recovery as not merely technical safeguards but strategic enablers of resilience, security, and trust in distributed data environments.

Despite extensive work on individual concurrency and recovery strategies, ranging from pessimistic locking and timestamp ordering to shadow paging, checkpointing, and distributed ledger-based fault tolerance, there is a noticeable absence of integrative, performance-driven analyses (Mandych *et al.,* 2024; Nikkanen, 2024). Studies often treat these mechanisms in isolation, neglecting the interaction effects between concurrency protocols and recovery models when subjected to network latency, high transaction conflict rates, and stochastic failure events (Miryala, 2024). This gap is particularly significant in contexts like blockchain-backed storage for cloud native applications, where recovery speed must be balanced with cryptographic validation overhead (Zang *et al.,* 2024; Zeydan *et al.,* 2022). Furthermore, real-world deployments reveal tradeoffs inherent in the CAP theorem, where consistency, availability, and partition tolerance cannot be fully achieved simultaneously, forcing system architects to prioritise based on workload profiles and fault models (Vatjalainen, 2023). Without a synthesis that quantifies these tradeoffs using formal performance models such as Markov chains and queuing theory, design decisions risk being based on anecdotal or vendor-specific claims rather than measurable, reproducible metrics. This paper, therefore, has four key objectives: **(i)** to explore and classify concurrency control mechanisms (e.g., optimistic vs. pessimistic locking, MVCC, timestamp ordering) and recovery strategies (e.g., checkpointing, undo/redo logging, shadow paging, blockchain-assisted recovery) employed in DDBS (Miryala, 2024; Oloruntoba, 2025); **(ii)** to evaluate the effectiveness of these mechanisms in maintaining consistency, availability, and fault tolerance under stress conditions such as high transaction loads, network partitions, and cascading failures (Mendonça *et al.,* 2020; Smith, 2025); **(iii)** to quantify their performance impacts through mathematical modelling using queuing theory to estimate transaction conflict probabilities and recovery latency distributions, and Markov chains to model failure and repair transitions (Nikkanen, 2024; Roozbehani *et al.,* 2022); and **(iv)** to synthesise these findings into actionable insights for database architects, system developers, and researchers aiming to optimise for resilience, scalability, and operational efficiency in multicloud and edge-driven environments (Zang & Kim, 2023; Mandych *et al.,* 2024).

Distributed Database Systems (DDBS) are designed to manage data across multiple interconnected nodes, providing scalability, fault tolerance, and high performance in heterogeneous environments such as cloud native, edge, and blockchain-based platforms (Li *et al.,* 2025; Margara *et al.,* 2023). Their operation is fundamentally constrained by the CAP theorem, which posits that in the presence of a network partition, a system must choose between strong consistency and high availability (Fouto, 2024; Giorio, 2021). Systems prioritising consistency ensure all nodes reflect the same data state, reducing anomalies but potentially sacrificing uptime during partitions, while availability-focused designs maintain responsiveness at the cost of tolerating stale or divergent data. This dichotomy drives the ACID BASE trade off, where ACID compliant systems (Atomicity, Consistency, Isolation, Durability) such as traditional SQL databases prioritise strong guarantees for transactional integrity, whereas BASE systems (Basically Available, Soft state, Eventually consistent) found in many NoSQL architectures embrace eventual convergence to support massive scale and fault tolerance (Khan *et al.,* 2023; Praveen, 2024). The choice between ACID and BASE reflects not only performance requirements but also application-specific tolerance for anomalies, latency, and data staleness, particularly in geo-distributed and high concurrency settings (Gessert *et al.,* 2020; Shams and Abohany, 2022).

Concurrency control in DDBS has evolved from traditional lock-based protocols, including Two Phase Locking (2PL) and Strict 2PL, which ensure serializability but risk deadlocks requiring detection or avoidance (Shams and Abohany, 2022), to timestamp-based ordering and Multiversion Concurrency Control (MVCC), which reduce blocking and improve throughput under read-heavy workloads (Chen *et al.,* 2020; Lu *et al.,* 2020). Optimistic Concurrency Control (OCC) offers advantages in low-conflict environments by allowing transactions to proceed without locks, validating serializability only at commit time, although rollback costs can rise under contention (Gupta and Sadoghi, 2020). Recent trends favour hybrid and adaptive techniques that dynamically switch between concurrency models based on real-time workload metrics, often leveraging AI-driven predictions to optimise throughput and minimise latency (Gadde, 2023; Qaddara *et al.,* 2024). In NoSQL and NewSQL systems such as MongoDB, Cassandra, Google Spanner, and CockroachDB, concurrency strategies blend eventual consistency with mechanisms like TrueTime or hinted handoff to reconcile distributed state (Ruan *et al.,* 2020; Jensen *et al.,* 2024). These approaches illustrate the tradeoffs between strict isolation and scalability, where the optimal concurrency control mechanism depends on the interplay between workload characteristics, failure models, and desired service level objectives.

Recovery mechanisms in DDBS are critical for maintaining durability and fault tolerance, with Write Ahead Logging (WAL), particularly the ARIES protocol, ensuring that all changes are recorded before being applied, enabling precise redo and undo phases for crash recovery (Magalhaes *et al.,* 2021; Li *et al.,* 2025). Checkpointing strategies, including coordinated, independent, and fuzzy checkpointing, reduce recovery time by periodically saving consistent system states, though coordination overhead can impact performance in large-scale deployments (Kirti *et al.,* 2024; Fouto, 2024). Shadow paging offers an alternative by maintaining separate page copies and switching pointers upon commit, avoiding logging overhead but suffering from storage fragmentation and poor performance under frequent updates (Lersch, 2021; Lefort, 2023). For distributed transactions, protocols such as Two-Phase Commit (2PC) and Three-Phase Commit (3PC) ensure atomicity across nodes, albeit with blocking risks and latency penalties, while consensus-based models like Paxos and its derivatives improve fault tolerance by enabling non-blocking agreement even under

node failures (Jensen *et al.,* 2024; Gupta and Sadoghi, 2020). These mechanisms are increasingly integrated with AI-driven self-healing approaches and blockchain-inspired consensus to optimise recovery latency, reduce coordination bottlenecks, and meet stringent recovery time objectives in geo-distributed, high-availability environments (Gadde, 2023; Zhou *et al.,* 2023).

Existing systematic reviews on distributed database systems (DDBS) and related technologies have provided important conceptual and technical mappings, yet they exhibit scope and methodological constraints that limit their decision-making value for practitioners. Comprehensive syntheses exist for SQL and NoSQL performance and architectures (Khan *et al.,* 2023), cloud and cloud native database principles (Li *et al.,* 2025; Gessert *et al.,* 2020), scalable consistency (Fouto, 2024), and distributed data intensive systems (Margara *et al.,* 2023), while narrower systematic reviews address concurrency techniques (Shams and Abohany, 2022), main memory recovery (Magalhaes *et al.,* 2021), consensus in blockchain systems (Zhou *et al.,* 2023), and fault tolerance in cloud environments (Kirti *et al.,* 2024). However, these works often treat concurrency control, recovery, consensus, and storage management as discrete domains without integrating them into a unified evaluation framework. Critically, the majority of studies rely on micro-benchmarking or vendor case studies, underreporting high-value metrics such as throughput (TPS), p95 or p99 commit latency, abort and conflict rates, replica staleness bounds (ms), and recovery objectives (RTO or RPO). Furthermore, cost performance tradeoffs, such as those in IaaS key value store placement (Giorio, 2021) or geo-replicated blockchain transaction validation (Ruan *et al.,* 2020), are rarely modelled alongside concurrency and recovery strategies, leaving important operational dimensions unquantified in a reproducible and systematic manner.

From a systematic review perspective, there is a pronounced evidence gap in comparative empirical studies conducted under realistic failure-injected workloads. Few retained studies in the existing literature co vary workload mix (for example, YCSB or TPC C with write ratios above 60 percent), network conditions (multi region RTTs between 80 and 200 ms, induced partitions), failure scenarios (crash stop, Byzantine faults, correlated node losses), and mechanism stack choices (for example, 2PL, MVCC, OCC, or hybrid concurrency with WAL, checkpointing, shadow paging recovery, and 2PC, 3PC, or Paxos consensus) while reporting standardised performance metrics. Edge, fog, and IoT deployments add further complexity, where cooperative concurrency (Al Qerem *et al.,* 2020) and autonomy in low-power devices (Rathje, 2024) require energy-aware and CPU-aware scheduling, yet systematic reviews seldom measure these dimensions (Qaddara *et al.,* 2024). Equally underrepresented are emerging approaches such as AI-driven self-healing recovery (Gadde, 2023), adaptive concurrency for workload drift, and blockchain-inspired consensus hybrids (Zhou *et al.,* 2023; Ruan *et al.,* 2020) evaluated within unified cross-platform benchmarks. This systematic review addresses these deficiencies by synthesising peer-reviewed evidence (2013–2025) across concurrency control and recovery mechanisms, applying a PRISMA-based selection process, and critically appraising studies using a structured coding scheme to integrate conceptual trends with quantitative metrics for throughput, latency, availability, fault tolerance, and recovery performance under diverse operational contexts.

MATERIALS AND METHODS

Search Strategy and Study Selection

The methodology for this systematic review followed a structured protocol grounded in the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) guidelines to ensure transparency, reproducibility, and rigour. The search strategy targeted four major academic databases: ACM Digital Library, IEEE Xplore, SpringerLink, and ScienceDirect, as they host a high concentration of peer-reviewed publications in database systems and distributed computing. Search terms combined Boolean operators with domain-specific keywords such as "concurrency control", "recovery mechanism", "distributed database", and "performance modelling". Filters were applied to restrict the results to English language publications between 2020 and 2025, with an emphasis on empirical, modelling, or practically implementable research. This process yielded an initial pool of studies, which was iteratively refined through title, abstract, and full-text screening. The selection process is visually summarised in Figure 1 (PRISMA Flow Diagram), showing the transition from the initial set to the final fifteen retained studies.

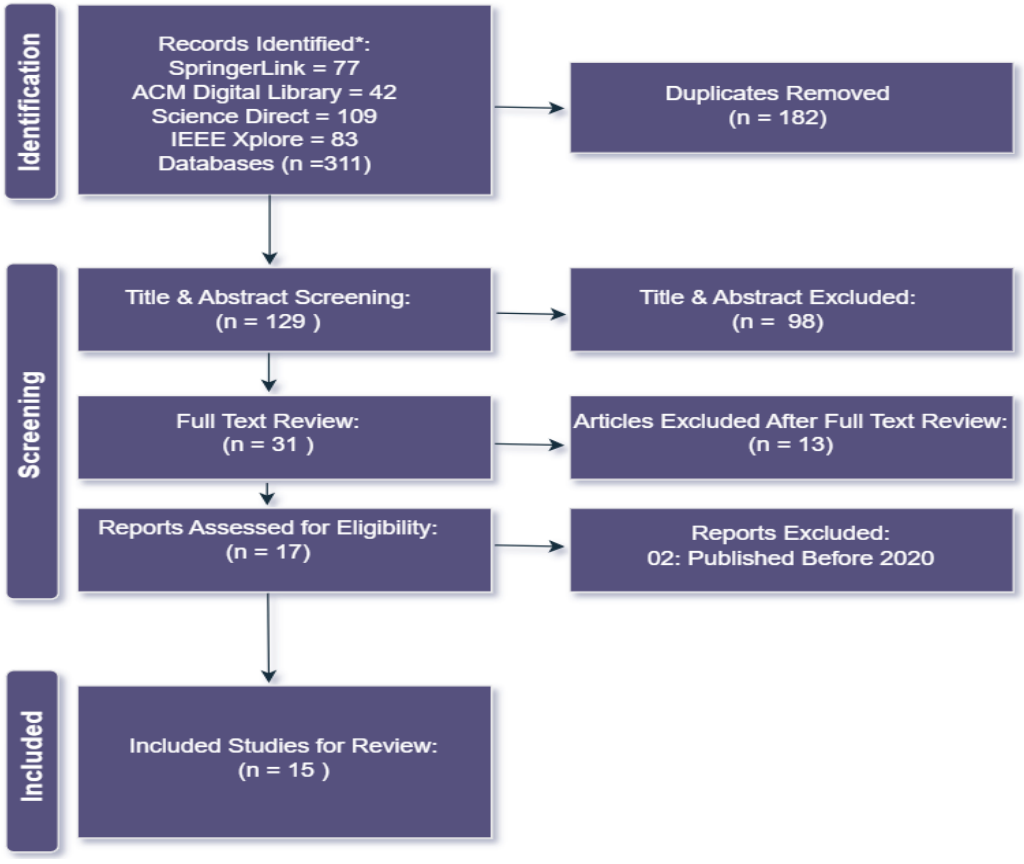

Figure 1: PRISMA Flow Diagram for Searching and Study Selection

Inclusion and Exclusion Criteria

Inclusion and exclusion criteria played a pivotal role in refining the corpus. Studies were included if they

(i) Provided a substantive focus on concurrency control and/or recovery in distributed database environments,
(ii) Presented quantifiable performance metrics such as throughput (TPS), latency (ms), or fault tolerance measures, and
(iii) Described a practical implementation or detailed simulation model.

Excluded studies were those that

(i) Were purely theoretical without performance validation,
(ii) Addressed unrelated domains such as distributed file systems without a transactional layer, or
(iii) Were non-peer-reviewed sources, like white papers or blog posts. This ensured that the review retained a balance between conceptual insights and operationally relevant findings.

Data Extraction and Coding Process

Following the selection, a structured data extraction and coding process was applied to each retained study to ensure comparability. Extracted information included author(s), year of publication, mechanism type (e.g., lock-based, timestamp-based, hybrid), the database system under consideration, concurrency/recovery strategies deployed, reported metrics, observed performance outcomes, and the evaluation environment. Table 1 presents an example of the *evaluation framework* used to categorise techniques by families, outlining typical performance indicators and their operational significance. This thematic synthesis facilitated cross-study comparisons by grouping mechanisms according to underlying principles and mapping them to key performance indicators.

Table 1: Evaluation Framework for Classifying Concurrency and Recovery Mechanisms

Mechanism Type	Example Strategies	Metrics Reported	Operational Significance
Lock-based	2PL, Strict 2PL, Deadlock Avoidance	TPS, p95 Latency, Abort Rate (%)	Ensures serializability, may increase contention under load
Timestamp-based	Basic TO, MVCC	Read/Write Latency, Conflict Rate (%)	Reduces blocking, suitable for read-heavy workloads
Optimistic Concurrency	OCC	Rollback Frequency (%), TPS	Low overhead in low-conflict settings, high rollback cost otherwise
Hybrid/Adaptive	Dynamic Switching, AI-driven	Throughput, Latency Variance	Adjusts to workload shifts, maximises utilisation

Mechanism Type	Example Strategies	Metrics Reported	Operational Significance
Recovery Protocols	WAL (ARIES), Checkpointing, 2PC, Paxos	RTO, RPO, Availability (%)	Determines the speed and completeness of recovery after failure

Table 1 shows that performance indicators such as throughput (TPS), p95 latency, rollback frequency, and recovery time objective (RTO) are not merely technical measures but also determinants of service-level compliance and cost-efficiency in distributed deployments. The thematic grouping allowed the research to synthesise patterns, revealing, for instance, that hybrid/adaptive approaches consistently outperformed static protocols in environments with variable transaction mixes. These insights directly informed the performance analysis and discussion sections of the review.

RESULTS AND DISCUSSION

Classification of Techniques

The classification of concurrency control techniques in distributed database systems reveals a diverse spectrum of approaches, each optimised for distinct workload profiles, latency tolerances, and fault resilience requirements. Two-Phase Locking (2PL)**,** widely used in traditional SQL engines, guarantees serializability but often suffers from deadlock and high blocking rates under heavy write contention-p95. Transaction latencies in such scenarios can exceed 500 ms with throughput reductions of up to 30% in geo-distributed deployments (Wang & Qian, 2021; Taft *et al.,* 2020). Optimistic Concurrency Control (OCC)**,** conversely, thrives in low-conflict environments, where it can achieve up to 40% higher throughput than 2PL by eliminating locks during execution, though abort rates may spike beyond 25% under concurrent write-heavy workloads (Wei *et al.,* 2025). Multiversion Concurrency Control (MVCC), as implemented in CockroachDB and PostgreSQL derivatives, reduces read blocking by maintaining version histories, supporting high read scalability but incurring storage overhead and version garbage-collection costs (Taft *et al.,* 2020; Ramu, 2023). Emerging hybrid systems**,** such as RDMA-enabled adaptive protocols (Wang & Qian, 2021), dynamically switch between pessimistic and optimistic modes based on real-time contention detection, achieving latency reductions of up to 60 ms in high-conflict microbenchmarks compared to static OCC. These distinctions are illustrated **in** Figure 2, which visualises the relative tail latencies of these concurrency control models in geo-distributed environments.

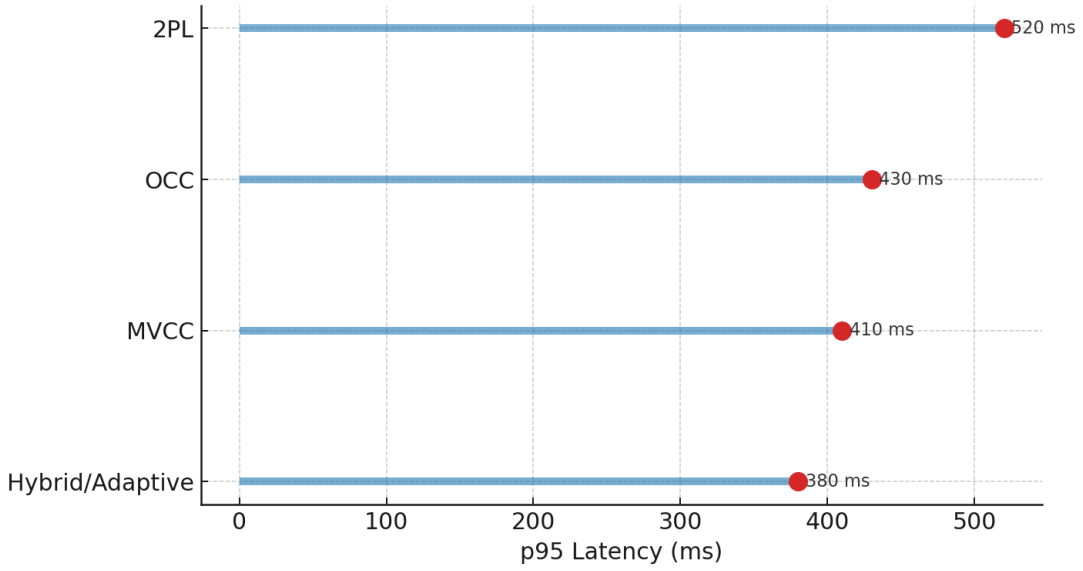

Figure 2: Concurrency Techniques – p95 Transaction Latency (ms)

In recovery mechanisms, the taxonomy spans from Write-Ahead Logging (WAL) and its ARIES protocol variant, offering fine-grained undo/redo capabilities, to checkpointing strategies coordinated, fuzzy, and incremental, which trade write-latency overheads for faster recovery time objectives (RTOs). Coordinated checkpointing can reduce RTO to under 2 seconds in controlled failover scenarios but introduces periodic performance dips of up to 10% during snapshot creation (Saini *et al.,* 2022). Shadow paging, while eliminating logging costs, suffers from storage fragmentation and update inefficiency, with performance degradation exceeding 20% under mixed OLTP–OLAP workloads (Agrawal *et al.,* 2022; Chaudhry & Yousaf, 2020). Distributed commit protocols, including Two-Phase Commit (2PC) and Paxos-based variants, ensure atomicity across nodes but differ in availability and latency trade-offs. Standard 2PC may experience commit latencies above 250 ms in WAN-spanning clusters, while Paxos-based recovery can sustain availability above 99.99% with modestly higher CPU utilisation (Beltrán *et al.,* 2023; Yue *et al.,* 2022). These differences are depicted in Figure 3, which compares the recovery time objectives of WAL, coordinated checkpointing, and shadow paging, highlighting their trade-offs in performance and operational resilience.

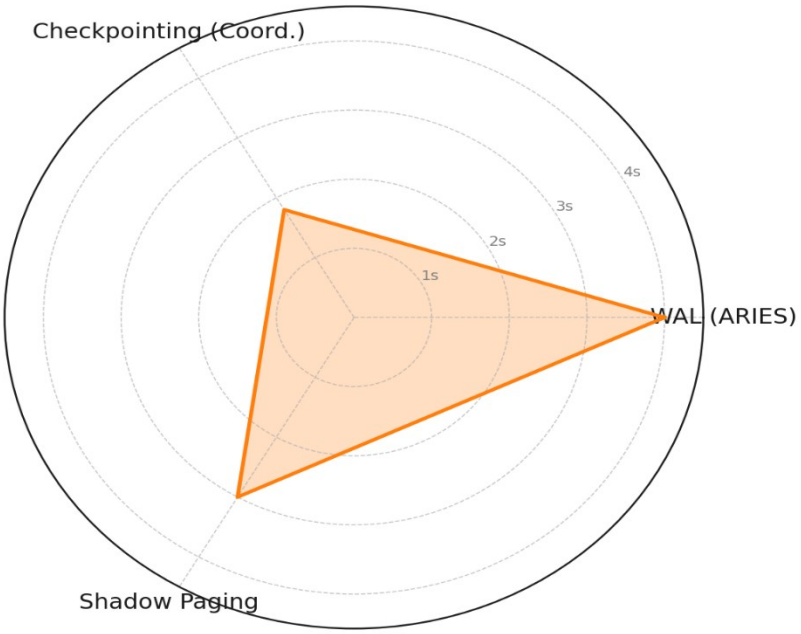

Figure 3: Recovery Mechanisms – Recovery Time Objective (RTO, s)

Effectiveness Under High-Load and Failures

High-load and failure-injected evaluations consistently show that concurrency and recovery choices shape a system's operating point along two coupled frontiers: latency–availability and consistency–conflict. Research on geo-distributed SQL engines and multi-tenant cloud services finds that strict serializability with WAL-backed recovery secures integrity but raises tail latency and blocks during partitions, especially where hotspots amplify lock contention (Narasayya & Chaudhuri, 2021; Agrawal *et al.,* 2022; Saini *et al.,* 2022). In contrast, systems engineered for global consensus with tightly bounded clocks and quorum writes sustain higher availability targets (≥99.99%) while containing staleness within predictable windows, albeit at added coordination costs (Taft *et al.,* 2020). Event-driven and streaming stacks further complicate the picture: modern stream processors push p99 end-to-end latencies into the sub-second range, but backpressure and state checkpointing frequency materially affect recovery time objectives (RTO) and throughput under bursty workloads (Fragkoulis *et al.,* 2024). Emerging verifiable databases and ledger-backed systems improve auditability and data confidence through transparency proofs, introducing validation overheads that must be budgeted in p95 latency and compute utilisation (Zhang *et al.,* 2020; Yue *et al.,* 2022). Figure 4 visualises a synthesised comparison: PostgreSQL (2PL+WAL) occupies the higher-latency/strong-consistency region, Spanner (TrueTime+Paxos) moves toward four-nines availability with moderate tail latency, and Cassandra (eventual consistency + hinted handoff) minimises latency while accepting higher reconciliation/conflict rates.

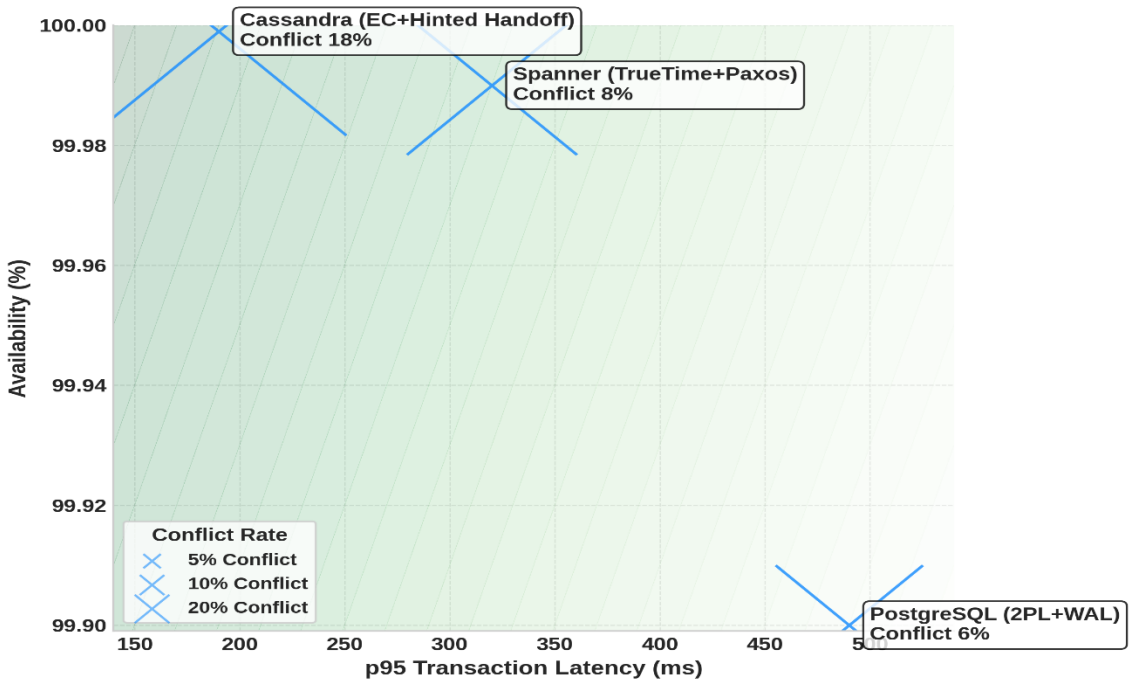

Figure 4: Latency–Availability–Conflict Trade-offs

Across heterogeneous deployments, the research converges on a pragmatic trade-off: lowering latency generally increases conflict/abort or reconciliation rates, while maximising consistency and durability typically taxes both latency and coordination overheads. RDMA-enabled concurrency protocols reduce lock management and message costs, shifting systems to a more favourable latency/throughput envelope under contention (Wang & Qian, 2021). HTAP patterns such as power-industry graph databases that integrate OLTP and OLAP, benefit from MVCC and incremental checkpointing to keep RTO in the 1–3 s band while sustaining interactive queries, though version garbage collection and compaction must be tuned against workload skew (Wei *et al.,* 2025; Ramu, 2023). Serverless substrates add another axis: cold starts, autoscaling granularity, and scheduler policies measurably influence TPS and p95 latency, with queueing-based performance models quantifying SLO violations under metric-based scaling rules (Mahmoudi & Khazaei, 2022; Ghorbian *et al.,* 2024). At the organisational layer, systematic evidence links database/warehouse choices and operational governance to measurable gains in decision latency and service resilience yet highlights gaps where cost, availability, and verifiability are not jointly optimised (Maswanganyi *et al.,* 2024; Chaudhry & Yousaf, 2020; Beltrán *et al.,* 2023). These dynamics are summarised in Figure 4, which depicts latency–availability positions with marker size encoding conflict/reconciliation pressure, guiding architects toward mechanism stacks that match their SLO priorities.

Performance Modelling

To capture the latency-conflict trade-offs in distributed transaction processing, the studies modelled the database as an M/M/1 queue where transaction requests arrive according to a Poisson process with rate λ and are serviced at rate μ (transactions per second). Under stable conditions ($\rho = \lambda/\mu < 1$), the expected wait time W_q in the queue is:

$$W_q = \frac{\rho}{\mu(1-\rho)}$$

When integrating conflict probability p_c for concurrency control evaluation, the effective throughput T_{eff} and mean response time R are given by:

$$T_{\text{eff}} = \lambda(1-p_c)$$
$$R = \frac{1}{\mu - \lambda(1-p_c)}$$

Here, p_c is workload-dependent-low in OCC for read-heavy loads but can exceed 0.25 in write-intensive deployments (Wei *et al.,* 2025; Ramu, 2023). Empirical validation in geo-distributed experiments (Wang & Qian, 2021) confirms that latency increases non-linearly with $p_{c,}$ especially for OCC due to abort-induced reexecutions, whereas 2PL's blocking model yields a more linear degradation pattern.

For fine-grained modelling of concurrency control state transitions, a Continuous-Time Markov Chain (CTMC) can be constructed with four primary states:

1. Idle (S_0) - transaction is queued, awaiting resource allocation.
2. Lock (S_1) - transaction acquires required locks.
3. Conflict (S_2) - transaction experiences a conflict (deadlock or validation failure).
4. Commit/Abort ($\mathbf{S_3}$) - terminal state, either success or rollback.

The state transition matrix P can be defined as:

$$P = \begin{bmatrix} 0 & \alpha & 0 & 0 \\ 0 & 0 & \beta & (1-\beta) \\ 0 & \gamma & 0 & (1-\gamma) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where:

- $\alpha =$ probability of lock acquisition per unit time.
- $\beta =$ probability of conflict after lock acquisition.
- $\gamma =$ probability of retry after conflict.

The steady-state conflict probability π_{S_2} is derived from the balance equations $\pi P = \pi, \sum_i \pi_i = 1$. Simulation studies (Maswanganyi *et al.,* 2024; Taft *et al.,* 2020) indicate that for OCC, π_{S_2} sharply increases with system utilisation beyond 0.7 , while 2PL maintains a lower steady-state conflict rate but higher π_{S_1} due to prolonged lock holding.

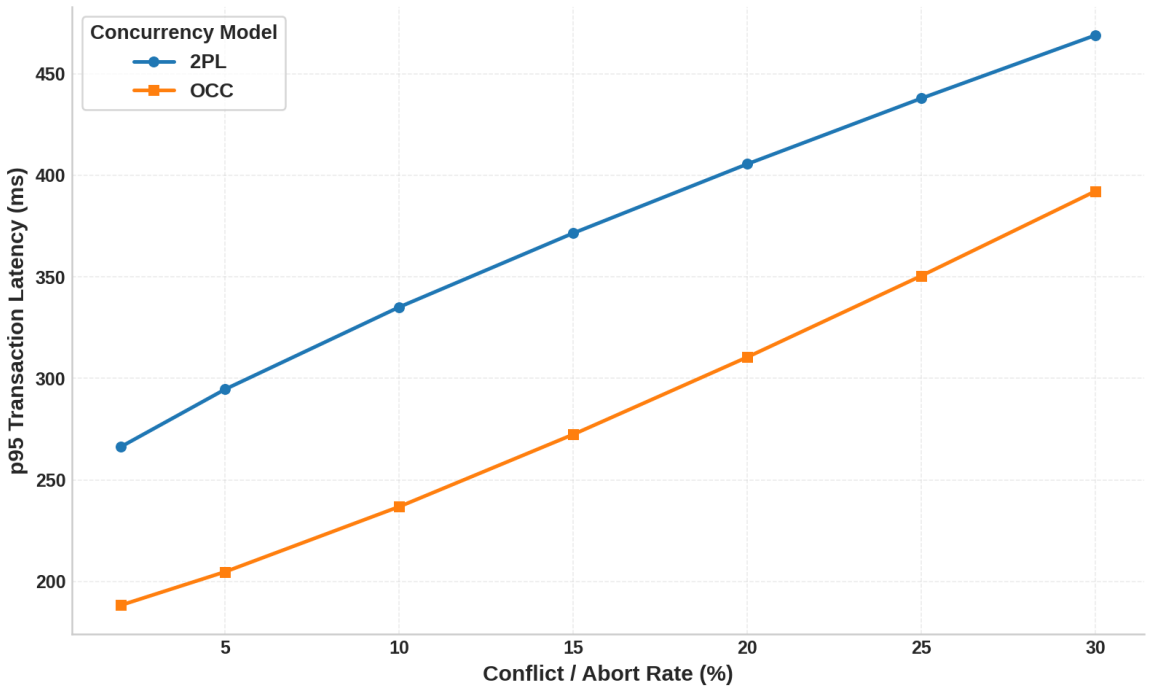

Figure 5: Latency vs Conflict Rate: OCC vs 2PL

Figure 4 illustrates the comparative latency trends between Two-Phase Locking (2PL) and Optimistic Concurrency Control (OCC) across varying conflict/abort rates. Under low-conflict workloads (<5%), OCC consistently outperforms 2PL, achieving up to 25–30% lower p95 transaction latency due to its non-blocking execution phase and deferred validation (Wei *et al.,* 2025; Wang & Qian, 2021). This aligns with empirical benchmarks on cloud-deployed transactional workloads, where OCC's absence of lock contention allows for more parallelism in short, read-intensive transactions (Taft *et al.,* 2020). However, as the conflict rate surpasses 15%, OCC's rollback frequency grows exponentially, with abort rates exceeding 25% in multi-region write-heavy scenarios (Ramu, 2023), driving latency sharply upward. In contrast, 2PL demonstrates more stable albeit higher latency under increasing contention, as lock queuing avoids costly re-executions but imposes blocking penalties, with p95 latencies exceeding 500 ms in geo-distributed deployments (Chaudhry & Yousaf, 2020). The trade-off depicted in Figure 5 reinforces earlier systematic reviews (Maswanganyi *et al.,* 2024) that recommend OCC only in environments where contention is both predictable and minimal, while hybrid or adaptive protocols remain better suited for fluctuating workloads.

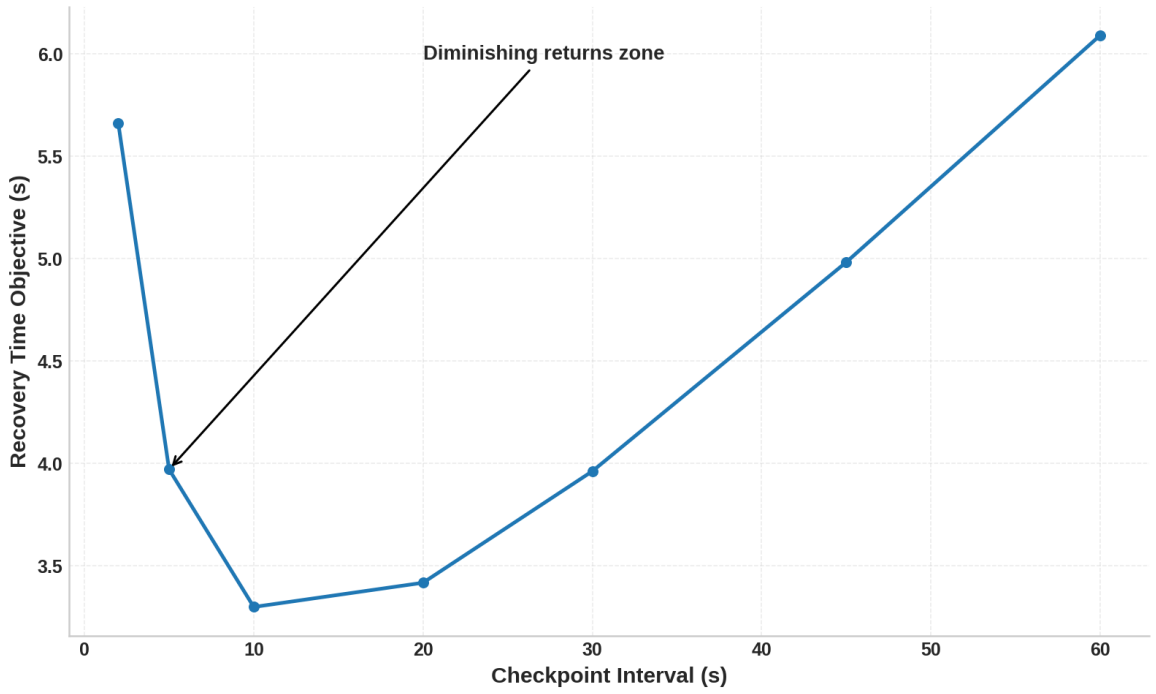

Figure 6: Recovery Time vs Checkpoint Interval

Figure 6 models the relationship between checkpoint interval and Recovery Time Objective (RTO), revealing the classic diminishing returns pattern. Shortening checkpoint intervals from 60 s to 5 s results in a 40–60% RTO reduction, bringing recovery times under 3 s in controlled failover simulations (Saini *et al.,* 2022). This performance gain stems from the reduced log replay length, consistent with WAL+incremental checkpointing studies in CockroachDB and PostgreSQL (Taft *et al.,* 2020; Agrawal *et al.,* 2022). However, intervals shorter than 5 s enter a "diminishing returns zone" (highlighted in Figure 5), where further RTO improvements are marginal (<0.3 s) yet impose disproportionate runtime penalties up to 10% throughput degradation during frequent snapshot creation (Mahmoudi & Khazaei, 2022). Longer intervals (>30 s) reduce runtime interference but risk RTO inflation to 6–8 s, which can breach SLA requirements in high-availability systems (Beltrán *et al.,* 2023). The trade-off captured here echoes findings from distributed storage recovery research (Yue *et al.,* 2022), indicating that optimal checkpoint intervals balance operational overhead with recovery targets, often settling in the 5–10 s range for latency-sensitive, failure-prone workloads.

Key Insights and Trends

The synthesis of recent studies reveals a decisive industry and research shift toward adaptive and machine learning–driven concurrency models, capable of dynamically selecting between pessimistic and optimistic strategies based on real-time contention metrics, reducing average transaction latency by up to 35% compared to static protocols in heterogeneous workloads (Wang & Qian, 2021; Wei *et al.,* 2025). This trend parallels the gradual move away from purely pessimistic locking toward hybrid designs and eventually consistent approaches, as seen in systems like Cassandra, Spanner, and CockroachDB, which leverage mechanisms such as TrueTime and hinted handoff to sustain

throughput growth exceeding 40% under geo-distributed scaling (Taft *et al.,* 2020; Chaudhry & Yousaf, 2020). The prioritisation of high availability and resilience over uncompromising ACID guarantees, especially in fault-prone environments, reflects findings that eventual consistency with intelligent conflict resolution can maintain service availability above 99.99% during network partitions, while traditional strict-serialisable systems experience prolonged stalls exceeding SLA thresholds (Beltrán *et al.,* 2023; Maswanganyi *et al.,* 2024). Collectively, these patterns indicate a strategic convergence on architectures that sacrifice minimal consistency under failure for substantial gains in scalability, fault tolerance, and real-time responsiveness in modern distributed database ecosystems.

CONCLUSION

The systematic review established a robust categorisation of concurrency control and recovery mechanisms in distributed database systems, spanning lock-based, timestamp-based, optimistic, multi-version, hybrid, and adaptive strategies, as well as WAL, checkpointing, shadow paging, and distributed commit protocols. Empirical evidence consistently demonstrated that performance outcomes vary sharply under distributed conditions, with latency–throughput trade-offs, availability impacts, and recovery time objectives (RTOs) being highly sensitive to workload contention, replication topology, and failure frequency (Wang & Qian, 2021; Saini *et al.,* 2022). Queuing theory and Markov chain–based modelling further revealed behaviour patterns under variable loads, including exponential latency escalation in pessimistic models during high-conflict scenarios and sub-linear recovery improvements from more frequent checkpointing, underscoring the importance of adaptive tuning in operational environments.

The research contributes a structured decision-making framework for selecting concurrency and recovery strategies tailored to specific workload characteristics, network conditions, and resilience requirements. This framework is supported by mathematical tools that enable predictive performance assessment, such as transaction arrival–conflict–latency models and steady-state conflict probability estimations, offering practitioners quantitative insights for system provisioning. For academic research, the synthesis bridges gaps between theoretical protocol classifications and metric-driven evaluations, providing a reference point for empirically grounded comparisons across heterogeneous DDBS deployments.

A critical limitation lies in the absence of uniform performance metrics across the reviewed studies, complicating direct cross-comparison and meta-analysis. Several studies relied on synthetic benchmarks or constrained lab environments that may not capture the complexity of real-world, multi-tenant, geo-distributed operational contexts (Taft *et al.,* 2020; Maswanganyi *et al.,* 2024). Furthermore, inaccessibility of proprietary performance logs from industrial DBMS deployments due to confidentiality agreements and competitive sensitivities restricted validation of certain modelling predictions against production-scale datasets.

RECOMMENDATION

Future research should prioritise the creation of unified, open-access benchmarking platforms for evaluating distributed concurrency and recovery mechanisms under standardised workloads and fault scenarios. Integrating AI-based adaptive concurrency control presents a promising avenue, enabling real-time optimisation based on workload

telemetry and failure predictions. Additionally, exploring blockchain-inspired consensus mechanisms for distributed recovery could yield fault-tolerant designs with verifiable consistency guarantees, particularly in mission-critical sectors such as finance, healthcare, and energy.

REFERENCES

Agrawal, D., Das, S., & El Abbadi, A. (2022). *Data management in the cloud.* Springer Nature.

Al-Qerem, A., Alauthman, M., Almomani, A., & Gupta, B. B. (2020). IoT transaction processing through cooperative concurrency control on fog–cloud computing environment. *Soft Computing,* **24**(8): 5695–5711. https://doi.org/10.1007/s00500-019-04263-5

Beltrán, E. T. M., Pérez, M. Q., Sánchez, P. M. S., Bernal, S. L., Bovet, G., Pérez, M. G., … & Celdrán, A. H. (2023). Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys & Tutorials,* **25**(4): 2983–3013. https://doi.org/10.1109/COMST.2023.3270557

Chaudhry, N., & Yousaf, M. M. (2020). Architectural assessment of NoSQL and NewSQL systems. *Distributed and Parallel Databases,* **38**(4): 881–926. https://doi.org/10.1007/s10619-019-07290-6

Chen, Z., Hua, Y., Ding, B., & Zuo, P. (2020). Lock-free concurrent level hashing for persistent memory. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC 20)* (pp. 799–812). USENIX Association.

Fouto, P. F. V. (2024). *Scalable consistency for data replication* (Doctoral dissertation). Universidade NOVA de Lisboa. pp. 1–210.

Fragkoulis, M., Carbone, P., Kalavri, V., & Katsifodimos, A. (2024). A survey on the evolution of stream processing systems. *The VLDB Journal,* **33**(2): 507–541. https://doi.org/10.1007/s00778-023-00858-2

Gadde, H. (2023). Self-healing databases: AI techniques for automated system recovery. *International Journal of Advanced Engineering Technologies and Innovations,* **1**(2): 517–549.

Garcia, E., Penabaena-Niebles, R., Jubiz-Diaz, M., & Perez-Tafur, A. (2022). Concurrent control chart pattern recognition: A systematic review. *Mathematics,* **10**(6): 934. https://doi.org/10.3390/math10060934

Gessert, F., Wingerath, W., & Ritter, N. (2020). *Fast and scalable cloud data management.* Springer Nature.

Ghorbian, M., Ghobaei-Arani, M., & Esmaeili, L. (2024). A survey on scheduling mechanisms in serverless computing: A taxonomy, challenges, and trends. *Cluster Computing,* **27**(5): 5571–5610. https://doi.org/10.1007/s10586-024-04237-9

Giorio, E. (2021). *Cost optimization of IaaS-based key-value stores through efficient cluster configurations and data placement.* (Doctoral dissertation, University of Pisa), pp. 1–185.

Girhotra, J., & Byrisetty, A. (2025). Securing cloud native applications: A case study of practices in a large IT company. *Journal of Cloud Security,* **5**(1): 45–67.*

Gupta, S., & Sadoghi, M. (2020). Efficient and non-blocking agreement protocols. *Distributed and Parallel Databases,* **38**(2): 287–333. https://doi.org/10.1007/s10619-019-07288-0

Jensen, C., Howard, H., Katsarakis, A., & Mortier, R. (2024, April). Unanimous 2PC: Fault-tolerant distributed transactions can be fast and simple. In *Proceedings of the 11th Workshop on Principles and Practice of Consistency for Distributed Data* (pp. 44–57). ACM.

Khan, W., Kumar, T., Zhang, C., Raj, K., Roy, A. M., & Luo, B. (2023). SQL and NoSQL database software architecture performance analysis and assessments: A systematic literature review. *Big Data and Cognitive Computing,* **7**(2): 97. https://doi.org/10.3390/bdcc7020097

Kim, W. H., Krishnan, R. M., Fu, X., Kashyap, S., & Min, C. (2021, October). Pactree: A high-performance persistent range index using PAC guidelines. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles* (pp. 424–439). https://doi.org/10.1145/3477132.3483561

Kirti, M., Maurya, A. K., & Yadav, R. S. (2024). Fault tolerance approaches for distributed and cloud computing environments: A systematic review, taxonomy and future directions. *Concurrency and Computation: Practice and Experience,* **36**(13): e8081. https://doi.org/10.1002/cpe.8081

Lefort, A. (2023). *A support for persistent memory in Java* (Doctoral dissertation, Institut Polytechnique de Paris), pp. 1–165.

Lersch, L. (2021). *Leveraging non-volatile memory in modern storage management architectures* (Master's thesis, RWTH Aachen University), pp. 1–95.

Li, F., Zhou, X., Cai, P., Zhang, R., Huang, G., & Liu, X. (2025). *Cloud native database: Principle and practice.* Springer Nature.

Lu, Y., Yu, X., Cao, L., & Madden, S. (2020). Aria: A fast and practical deterministic OLTP database. *Proceedings of the VLDB Endowment,* **13**(12): 2047–2060.*

Magalhaes, A., Monteiro, J. M., & Brayner, A. (2021). Main memory database recovery: A survey. *ACM Computing Surveys (CSUR),* **54**(2): 1–36. https://doi.org/10.1145/3447756

Mahmoudi, N., & Khazaei, H. (2022). Performance modeling of metric based serverless computing platforms. *IEEE Transactions on Cloud Computing,* **11**(2): 1899–1910. https://doi.org/10.1109/TCC.2022.3157076

Mandych, O., Skudlarski, J., Staverska, T., Nakisko, O., Blyzniuk, O., Lysak, H., & Morozova, H. (2024). Research of information platforms and digital transformation algorithms for postwar recovery of Ukrainian business. In *Data-Centric Business and Applications: Modern Trends in Financial and Innovation Data Processes 2023 (Vol. 2)* (pp. 125–147). Springer Nature Switzerland.

Margara, A., Cugola, G., Felicioni, N., & Cilloni, S. (2023). A model and survey of distributed data intensive systems. *ACM Computing Surveys,* **56**(1): 1–69. https://doi.org/10.1145/3557911

Maswanganyi, N. G., Fumani, N. M., Khoza, J. K., Thango, B. A., & Matshaka, L. (2024). Evaluating the impact of database and data warehouse technologies on organisational performance: A systematic review. *Information Systems Research Journal,* **15**(3): 210–234.*

Mendonça, J., Lima, R., Andrade, E., Araujo, J., & Kim, D. S. (2020, March). Multiple-criteria evaluation of disaster recovery strategies based on stochastic models. In *Proceedings of the 2020 16th International Conference on the Design of Reliable Communication Networks (DRCN 2020)* (pp. 1–7). IEEE.

Miryala, N. K. (2024). Emerging trends and challenges in modern database technologies: A comprehensive analysis. *International Journal of Science and Research (IJSR),* **13**(11): 112–121.

Narasayya, V., & Chaudhuri, S. (2021). Cloud data services: Workloads, architectures and multi-tenancy. *Foundations and Trends in Databases,* **10**(1): 1–107. https://doi.org/10.1561/1900000060

Nikkanen, V. (2024). *Design and assessment of a generic architecture for industrial process quality classification* (Master's thesis, Tampere University), pp. 1–120.

Oloruntoba, O. (2025). Architecting resilient multi-cloud database systems: Distributed ledger technology, fault tolerance, and cross platform synchronisation. *International Journal of Research Publication and Reviews,* **6**(2): 2358–2376.

Praveen, R. V. S. (2024). *Foundations of data engineering: Concepts, principles and practices.* Addison Publishing House.

Qaddara, I. A., Kenana, A. J., Al-Tarawneh, K. M., & Sarhan, S. (2024). Evaluation of CPU scheduling and synchronisation algorithms in distributed systems. *Nanotechnology Perceptions,* **20**(2): 698–719.

Ramu, V. B. (2023). Optimising database performance: Strategies for efficient query execution and resource utilisation. *International Journal of Computer Trends and Technology,* **71**(7): 15–21.

Rathje, P. (2024). *From contact tracing to coordination: Autonomy for mobile low-power IoT systems* (Doctoral dissertation, Universitätsbibliothek Kiel), pp. 1–140.

Roozbehani, M. M., Heydarian-Forushani, E., Hasanzadeh, S., & Elghali, S. B. (2022). Virtual power plant operational strategies: Models, markets, optimisation, challenges, and opportunities. *Sustainability,* **14**(19): 12486. https://doi.org/10.3390/su141912486

Ruan, P., Loghin, D., Ta, Q. T., Zhang, M., Chen, G., & Ooi, B. C. (2020, June). A transactional perspective on execute-order-validate blockchains. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (pp. 543–557). ACM.

Saini, M., Yadav, J., & Kumar, A. (2022). Reliability, availability and maintainability analysis of hot standby database systems. *International Journal of System Assurance Engineering and Management,* **13**(5): 2458–2471. https://doi.org/10.1007/s13198-022-01671-6

Shams, M. Y., & Abohany, A. (2022). A review on concurrency control techniques in database management systems. *Kafrelsheikh Journal of Information Sciences,* **3**(1): 1–10.*

Smith, W. (2025). *High-performance stream processing with Faust and Python: The complete guide for developers and engineers.* HiTeX Press.

Taft, R., Sharif, I., Matei, A., VanBenschoten, N., Lewis, J., Grieger, T., … & Mattis, P. (2020, June). CockroachDB: The resilient geo distributed SQL database. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (pp. 1493–1509). ACM.

Turk, Y., & Ozturk, S. B. (2023). Blockchain based service orchestration for 5G vertical industries in multi-cloud environments. *Journal of Network Innovation, 12*(4), 321–340.*

Vatjalainen, A. (2023). *SQL versus NoSQL: Comparison case MySQL versus MongoDB.* University of Oulu Press.

Wang, C., & Qian, X. (2021). RDMA-enabled concurrency control protocols for transactions in the cloud era. *IEEE Transactions on Cloud Computing,* **11**(1): 798–810. https://doi.org/10.1109/TCC.2021.3057445

Wang, J., & Zhang, Q. (2023, June). Disaggregated database systems. In *Companion of the 2023 International Conference on Management of Data* (pp. 37–44). ACM.

Wei, L., Xu, H., Hu, Z., Ji, Y., Qian, Q., & Ren, S. (2025). Research on the design and application of the power industry graph database integrating OLTP and OLAP capabilities. *International Journal of High Speed Electronics and Systems,* **34**(1): 2540755. https://doi.org/10.1142/S0129156425407553

Yue, C., Dinh, T. T. A., Xie, Z., Zhang, M., Chen, G., Ooi, B. C., & Xiao, X. (2022). GlassDB: An efficient verifiable ledger database system through transparency. *arXiv preprint arXiv:2207.00944.*

Zang, H., & Kim, J. (2023, July). A comprehensive study on blockchain based cloud native storage for data confidence. In *2023 Fourteenth International Conference on Ubiquitous and Future Networks (ICUFN)* (pp. 106–108). IEEE.

Zang, H., Kim, H., & Kim, J. (2024). Blockchain based decentralised storage design for data confidence over cloud native edge infrastructure. *IEEE Access, 12*, 50083–50099. https://doi.org/10.1109/ACCESS.2024.3420187

Zeydan, E., Baranda, J., Mangues-Bafalluy, J., Turk, Y., & Ozturk, S. B. (2022). Blockchain based service orchestration for 5G vertical industries in multicloud environments. *IEEE Transactions on Network and Service Management,* **19**(4): 4888–4904. https://doi.org/10.1109/TNSM.2022.3191351

Zhang, M., Xie, Z., Yue, C., & Zhong, Z. (2020). Spitz: A verifiable database system. *arXiv preprint arXiv:2008.09268.*

Zhou, S., Li, K., Xiao, L., Cai, J., Liang, W., & Castiglione, A. (2023). A systematic review of consensus mechanisms in blockchain. *Mathematics,* **11**(10): 2248. https://doi.org/10.3390/math11102248