Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Experimental Characterization of Metadata Synchronization in a Homogeneous MySQL Multidatabase Testbed Direct JSON-over-TCP/IP Fan-out, Serial JSON-over-SMB Relay, and Native Replication

Mohammad Ghulam Ali Ali

ali@hijli.iitkgp.ac.in

Indian Institute of Technology Kharagpur

Research Article

Keywords: metadata synchronization, multidatabase systems, DDL propagation, distributed catalogs, MySQL, TCP/IP, SMB, latency, schema consistency

Posted Date: October 30th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11451908/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Experimental Characterization of Metadata Synchronization in a Homogeneous MySQL Multidatabase Testbed

Direct JSON-over-TCP/IP Fan-out, Serial JSON-over-SMB Relay, and Native Replication

Mohammad Ghulam Ali, Member ACM SIGMOD, IAENG, ADU, GEP, Senior Member IACSIT
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur- 721302, INDIA
https://orcid.org/0000-0002-6012-6380

CORRESPONDING AUTHOR: M. G Ali (e-mail: ali@hijli.iitkgp.ac.in).

Abstract

Schema changes in autonomous database nodes can invalidate integrated catalog views unless metadata propagation is timely and correct. This study comparatively evaluates three operationally distinct Data Definition Language (DDL) propagation implementations in a three-node homogeneous MySQL environment: application-level direct TCP/IP fan-out, serial JSON-over-SMB relay, and native MySQL replication. Five latency observations per destination and technique yield 30 recorded measurements. The mean propagation latencies for Nodes 1 and 2 within same subnet is 1200.0 and Nodes 1 and 3 within two different subnets is 1383.0 ms for direct TCP/IP fan-out, Nodes 1 and 2 within same subnet is 3241.0 and Nodes 1 and 3 within two different subnets is 4664.6 ms for serial JSON-over-SMB relay, and Nodes 1 and 2 within same subnet is 171.6 and Nodes 1 and 3 within two different subnets is 146.4 ms for native MySQL replication. The comparatively higher serial-relay latency is consistent with cumulative processing and coordination overhead associated with sequential forwarding, JSON serialization, and SMB-based file handling. The study distinguishes metadata delivery, DDL application, acknowledgement, and catalog convergence as separate operational endpoints, examines descriptive latency variability and event-index comparisons, and establishes a catalog-consistency specification. The recorded observations show lower mean delays for direct TCP/IP fan-out than for serial relay, while native MySQL replication exhibits the lowest mean delays under the evaluated conditions. Nevertheless, architectural differences and measurement semantics limit direct causal attribution of performance differences to individual transport mechanisms. The findings provide empirical insights into the performance characteristics and architectural trade-offs of DDL propagation strategies, establishing a foundation for designing reliable metadata synchronization mechanisms in distributed database environments without claiming experimentally verified heterogeneous interoperability or global catalog convergence.

Keywords: metadata synchronization; multidatabase systems; DDL propagation; distributed catalogs; MySQL; TCP/IP; SMB; latency; schema consistency

1 Introduction

A multidatabase system coordinates independently managed databases while preserving local autonomy. A global view of metadata depends on the accuracy and freshness of local schema descriptions, including tables, attributes, keys and constraints [9,10]. A schema change can therefore create a temporal discrepancy between the source catalog, remote catalogs and a global integration layer. Prior work establishes the motivation for federation and schema integration [9–11]; the present experiment examines the narrower engineering question of propagation delay.

The measured environment comprises three MySQL nodes, not heterogeneous database engines. We compare two custom application-level DDL delivery paths with a native replication comparator. Unlike a controlled factorial experiment designed to isolate individual transport effects, the evaluated implementations differ simultaneously in network topology, event handling, storage mechanisms, serialization procedures, and completion-detection semantics. Consequently, the present study characterizes the observed performance of complete DDL propagation implementations rather than attributing latency differences to individual communication protocols.

1.1 Conceptual Classification of Related Research

Existing research relevant to metadata synchronization in multidatabase environments can be organized into four interrelated categories: (i) metadata maintenance and schema modification propagation, (ii) homogeneous/heterogenous database integration and schema modeling, (iii) distributed synchronization and replication mechanisms, and (iv) foundational distributed database principles.

1.1.1 Metadata Maintenance and Schema Modification Propagation

Research addressing metadata maintenance and local schema modification propagation provides the closest conceptual foundation for the present investigation. Ali [17] examined the propagation of local schema modifications and maintenance of metadata in multidatabase systems, while Ali [15] investigated an object-oriented approach to heterogeneous database integration and schema modification propagation. Subsequently, Ali [3] proposed replication-agent instances for maintaining metadata across multidatabase environments. SYNC SCHEMA operations [8] provide an additional implementation-oriented reference for schema synchronization in federated distributed databases. These contributions establish the importance of maintaining consistency between independently managed database schemas and integrated metadata representations. The present study extends this research direction by comparatively characterizing three operational DDL propagation implementations and distinguishing observed propagation latency from verified catalog convergence.

1.1.2 Heterogeneous Database Integration and Schema Modeling

A second research category concerns the integration of distributed databases and the construction of unified schema representations. Ali [13] investigated global schema construction using global views, while Ali [14] proposed proxy-table-based schema integration. Ali [16] described a four-tier client-server architecture for heterogeneous multidatabase systems, and Ali and Murtuza [12] discussed Component Integration Services as an approach to multidatabase integration. Fernández Candel et al. [6] developed a unified metamodel for relational and NoSQL databases, whereas Martins et al. [5] investigated schema references in keyword queries over relational databases. These studies primarily address schema representation, database integration, architectural organization, or schema-aware querying. In contrast, the present study focuses primarily on the operational propagation of DDL changes among homogeneous MySQL nodes, while also introducing an Apache Calcite-based federated query graphical user interface (GUI) as a conceptual integration component. The GUI provides an architectural interface for federated query access, whereas the experimental evaluation is confined to DDL propagation latency and does not independently assess federated query performance or heterogeneous schema transformation.

1.1.3 Distributed Synchronization and Replication Mechanisms

A third category encompasses synchronization mechanisms and replicated data management. Tambe [1] addressed bidirectional synchronization and conflict resolution for heterogeneous enterprise content platforms using vector clocks and semantic merge policies. Mondal and Tilevich [2] investigated cross-platform integration of replicated data libraries. Jasny et al. [4] examined synchronization of disaggregated data structures using one-sided RDMA, while Giannoula et al. [7] introduced synchronization support for near-data-processing

architectures. Although these investigations address different synchronization targets and computational environments, they provide useful comparative perspectives on coordination, replication, and synchronization performance. The present study differs by concentrating on database DDL propagation and comparing application-level TCP/IP fan-out, serial JSON-over-SMB relay, and native MySQL replication using destination-specific latency observations.

1.1.4 Foundational Distributed Database Principles

The foundational work of Sheth and Larson [9] established the architectural principles of federated database systems involving heterogeneous and autonomous databases. Özsu and Valduriez [10] provided theoretical foundations for distributed database management, replication, and consistency, while Elmasri and Navathe [11] discussed database schemas, metadata catalogs, and distributed database concepts. These contributions establish the theoretical context for understanding local autonomy, distributed catalog maintenance, and consistency requirements. The present investigation applies these concepts to a bounded experimental setting involving three homogeneous MySQL nodes and a formal specification of catalog-consistency conditions.

1.1.5 Research Synthesis and Distinctive Contribution

Collectively, the reviewed studies address database federation, global schema construction, metadata maintenance, synchronization architectures, and distributed consistency. The present research contributes a comparative empirical characterization of three DDL propagation implementations—direct JSON-over-TCP/IP fan-out, serial JSON-over-SMB relay, and native MySQL replication—in a homogeneous three-node MySQL testbed. Thirty recorded latency observations support destination-specific and descriptive variability comparisons. The study also distinguishes event delivery, DDL application, acknowledgement, and catalog-state consistency as separate criteria. Table 1 positions these contributions relative to the cited literature. Both custom workflows incorporate reliability mechanisms: direct TCP/IP fan-out uses destination execution ACKs and retry when ACKs are missing, whereas serial SMB relay includes failure detection, logging, retry, replay, dead-letter handling, and recovery. The Apache Calcite-based federated query GUI is an architectural integration component, not a benchmarked subsystem.

Table 1. Comparative matrix of related studies and the present research

Ref.	What the referenced work has done	What the present manuscript has done	Distinct approach / research differentiation
[1] Tambe (2026)	Describes bidirectional synchronization for heterogeneous enterprise content platforms, emphasizing vector clocks and semantic conflict-resolution policies,	Evaluates DDL propagation latency across three MySQL nodes using three implementation strategies.	Conflict resolution for replicated content versus propagation performance and metadata-consistency specifications for database schema changes.
[2] Mondal & Tilevich (2026)	Introduces BabelRDL, a common-data-format and plugin-based architecture for cross-platform replicated data libraries, evaluated for latency, memory usage, and throughput.	Compares direct TCP/IP, serial JSON-over-SMB, and native MySQL replication for DDL propagation.	Cross-platform replicated-library interoperability versus schema-change propagation within a homogeneous DBMS.
[3] Ali (2025)	Proposes replication-agent instances for maintaining metadata in a multidatabase system,	Empirically compares three operational DDL propagation implementations and their observed latencies.	Replication-agent-based metadata maintenance versus comparative latency characterization and explicit propagation endpoint semantics.
[4] Jasny et al. (2025)	Investigates correctness, performance, and scalability of synchronization mechanisms for disaggregated data structures using one-sided RDMA.	Investigates propagation of schema modifications among autonomous MySQL nodes.	Concurrent memory/data-structure synchronization versus database DDL propagation and catalog-consistency requirements.
[5] Martins et al. (2023)	Develops LATHE for keyword queries referring to relational schema elements,	Studies how schema changes are propagated to destination nodes.	Schema-aware information retrieval versus operational schema-change synchronization.

	using query matching, ranking, and join-network generation.		
[6] Fernández Candel et al. (2022)	Develops U-Schema, a unified metamodel for representing relational and several NoSQL schema paradigms.	Evaluates DDL propagation within a homogeneous MySQL environment.	Cross-model schema representation versus experimental propagation of schema modifications.
[7] Giannoula et al. (2021)	Introduces SynCron for synchronization support in near-data-processing architectures.	Evaluates distributed DDL propagation latency and distinguishes synchronization endpoints.	Hardware-oriented synchronization coordination versus database-level schema-change dissemination.
[8] SYNC SCHEMA	Provides schema import, definition comparison, and DDL synchronization operations for federated distributed databases.	Compares three propagation implementations and specifies catalog-consistency conditions.	Product-specific schema synchronization functionality versus controlled-scope empirical comparison of alternative DDL propagation implementations.
[9] Sheth & Larson (1990)	Establishes foundational concepts and architectures for federated databases, including heterogeneity, autonomy, and schema integration.	Investigates schema-change propagation performance among autonomous homogeneous nodes.	Foundational federation architecture versus implementation-level latency measurements.
[10] Özsu & Valduriez (2020)	Provides theoretical foundations for distributed database design, replication, consistency, and distributed processing.	Measures propagation delays and formulates endpoint-specific catalog-consistency requirements.	General distributed database theory versus a specific empirical DDL synchronization investigation.
[11] Elmasri & Navathe (2022)	Explains database schemas, metadata catalogs, distributed databases, and related database-management principles.	Examines propagation of schema modifications and their implications for metadata consistency.	Foundational database concepts versus experimental evaluation of DDL propagation.
[12] Ali & Murtuza (2023)	Discusses Component Integration Services as an approach to integrating databases in multidatabase environments.	Compares three approaches for disseminating schema modifications in MySQL.	Database integration through middleware services versus DDL propagation latency and synchronization semantics.
[13] Ali (2010)	Proposes global schema creation through global views over distributed heterogeneous relational databases.	Evaluates mechanisms relevant to maintaining schema information following local changes.	Global schema construction through views versus propagation of subsequent schema modifications.
[14] Ali (2010)	Proposes global schema creation through proxy tables over heterogeneous relational databases.	Investigates alternative mechanisms for propagating DDL changes among database nodes.	Proxy-table-based global schema integration versus empirical DDL propagation comparison.
[15] Ali (2009)	Investigates object-oriented heterogeneous database integration and local schema modification propagation.	Compares three concrete propagation implementations with recorded destination latencies.	Object-oriented integration and propagation framework versus implementation-level comparative evaluation.
[16] Ali (2009)	Describes a four-tier client-server architecture for a distributed heterogeneous multidatabase system.	Examines DDL propagation behavior in a three-node homogeneous MySQL testbed.	Multidatabase architectural organization versus measured synchronization performance.
[17] Ali (2010)	Addresses local schema modification propagation and metadata maintenance in multidatabase systems.	Extends the investigation toward comparative DDL propagation latency, event-index analysis, and catalog-consistency specifications.	Metadata maintenance framework versus quantitative comparison of three propagation mechanisms and formal separation of operational endpoints.

1.2 Novelty

The novelty of this study lies in the implementation-level comparison of three operationally distinct DDL propagation paths in a homogeneous multidatabase testbed. It combines destination-specific latency measurements, descriptive dispersion and event-index analyses, a topology-explicit distinction between direct fan-out and serial relay, and a formal global/local catalog-consistency specification. The custom approaches use existing TCP/IP and SMB mechanisms rather than introducing new transport protocols. Their ACK, retry, and recovery provisions are examined as architectural features, while measured performance is attributed to the complete implementations rather than to the transport mechanisms alone.

1.3 Research questions and contributions

RQ1: What delays and dispersion were recorded for each implementation and destination?

RQ2: How do direct fan-out and serial relay differ at the farther destination?

RQ3: Which observed differences are consistent with topology, and what remains unidentified?

RQ4: What correctness conditions would be required to elevate DDL propagation into verified catalog synchronization?

The contributions are (i) preservation and analysis of 30 recorded latency observations; (ii) an explicit comparison of direct and relay-dependent propagation paths; (iii) separation of delivery, application, acknowledgement, and catalog-consistency endpoints; (iv) descriptive aggregate and event-index comparisons; and (v) a formal metadata-consistency specification. The global-catalog model defines the intended relationship between the integrated and participating local metadata states; the reported experiment measures synchronization delays rather than independently validating that relationship.

2 Homogeneous multidatabase architecture and metadata catalogs

Each node in the three-node testbed runs MySQL and maintains a local system catalog. In the described architecture, a DDL operation is submitted through the global metadata-catalog coordination layer, captured and recorded at Node 1, and propagated to the required destination nodes. Destination execution, acknowledgement handling, and corresponding updates to local distributed catalogs and the global metadata catalog form the intended synchronization workflow. The Apache Calcite-based federated query GUI is included as an integration component for a unified query interface; it is not independently benchmarked in this study.

The two custom implementations follow different process flows. Direct JSON-over-TCP/IP fan-out sends the DDL event independently from Node 1 to Nodes 2 and 3, records destination execution ACKs, and retries when required ACKs are not received. Serial JSON-over-SMB queues and serializes the event, transfers it to Node 2 for execution, and then relays it to Node 3; its documented flow includes an ACK chain, failure logging, retry, replay, dead-letter handling, and recovery. The global metadata log coordinates the intended catalog-state updates in both approaches.

The experimental results report destination-specific synchronization delays for these complete workflows and for native MySQL replication. They do not separately quantify global-catalog refresh time, independently verify all resulting local catalog states, or measure federated-query GUI performance.

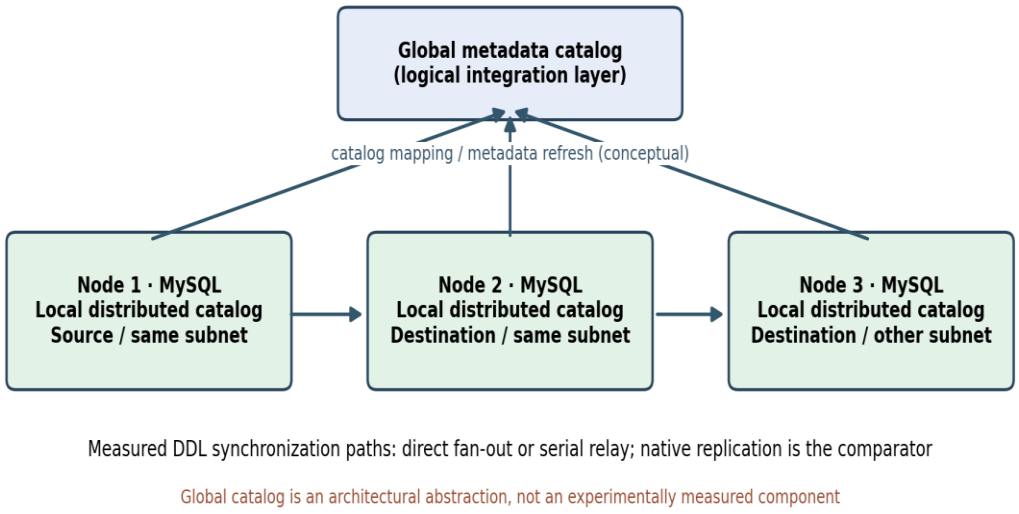


Figure 1. *Conceptual homogeneous multidatabase: three local MySQL catalogs and an integrated global metadata catalog (global update not measured).*

2.1 Formal catalog state and consistency criterion

Let $L_j(t)$ denote the local metadata catalog at node j at time t, for j = 1, …, n. Let M_j map local catalog descriptors into a normalized global representation. The global metadata log is intended to record schema-change events, whereas the global catalog represents the integrated schema state.

$$G*(t) = \cup_{j=1}^{n} M_j\left(L_j(t)\right) \tag{1}$$

Here the union denotes logical integration of normalized descriptors subject to a defined mapping and conflict-resolution policy; it is not an unrestricted union of conflicting schema definitions.

$$G(t^c) \equiv G*(t^c) \tag{2}$$

$G(t^c)$ is the maintained global catalog, and $G^*(t^c)$ is the expected integrated catalog. The equivalence relation ≡ means semantic agreement of relevant schema objects, attributes, types, keys, and constraints under the adopted normalization and reconciliation policy.

$$A_{ij} = \begin{cases} 1, & if\ DDL\ event\ e_i\ is\ successfully\ applied\ at\ node\ j \\ 0, & otherwise. \end{cases} \tag{3}$$

$$E_{ij} = \begin{cases} 1, & if\ N(L_j^{after\ e_i}) = N(L_{ij}^{expected}) \\ 0, & otherwise. \end{cases} \tag{4}$$

N denotes canonicalization of schema descriptors; L_j after e_i is the observed destination schema state and L_{ij} expected is the schema state expected after the applicable event. The success indicator $A_{ij} = 1$ does not, by itself, imply $E_{ij} = 1$ or global convergence.

Equations (1)–(4) specify the intended metadata-state and event-application correctness conditions. They are formal criteria rather than independently verified outcomes of the reported latency measurements.

3 Synchronization implementations

The study compares direct JSON-over-TCP/IP fan-out, serial JSON-over-SMB relay, and native MySQL replication. The two custom workflows differ in delivery topology, event handling, ACK coordination, and

recovery provisions; the comparison concerns the complete implementations rather than isolated network protocols.

3.1 Direct JSON-over-TCP/IP one-to-N fan-out

Node 1 captures and logs a DDL event (T0) and transmits it independently to Nodes 2 and 3 over TCP socket connections. The process flow identifies destination receipt (T1), execution completion (T2), ACK arrival (T3), and subsequent completion/consistency checking (T4). Missing destination ACKs trigger retry handling. This is application-level one-to-N fan-out, not network-layer broadcast; Node 2 shares the source subnet and Node 3 is on a different subnet. The recorded delay values do not separately resolve T0–T4, and no crash-and-restart experiment was reported to verify exactly-once execution under every retry condition.

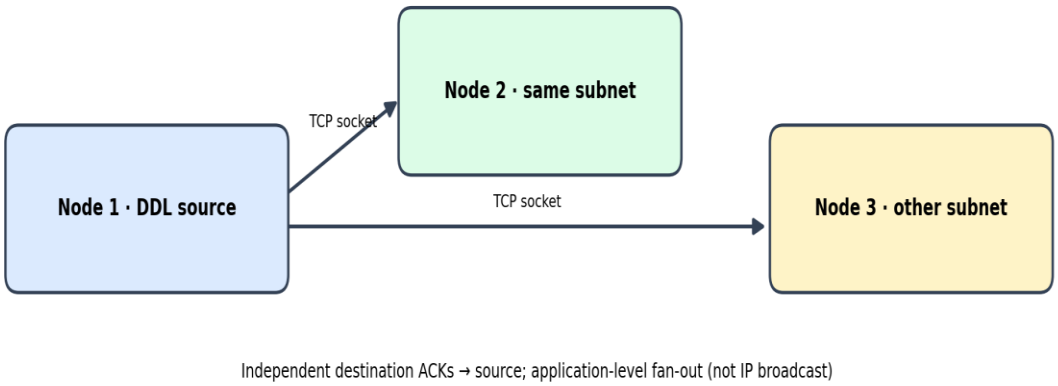


Figure 2. *Direct application-level TCP fan-out from Node 1 to both destinations; paths do not relay through Node 2.*

3.2 Serial JSON-over-SMB relay

In the serial JSON-over-SMB workflow, Node 1 captures, queues, and serializes a DDL event for file-based transfer to Node 2. Node 2 executes the operation before forwarding the event to Node 3, where it is also applied. The documented process flow includes relay-dependent acknowledgements, failure detection and logging, retry, replay, dead-letter handling, and recovery procedures. These are features of the specified workflow; the reported latency observations do not isolate their individual timing contributions or independently validate recovery under injected failures.

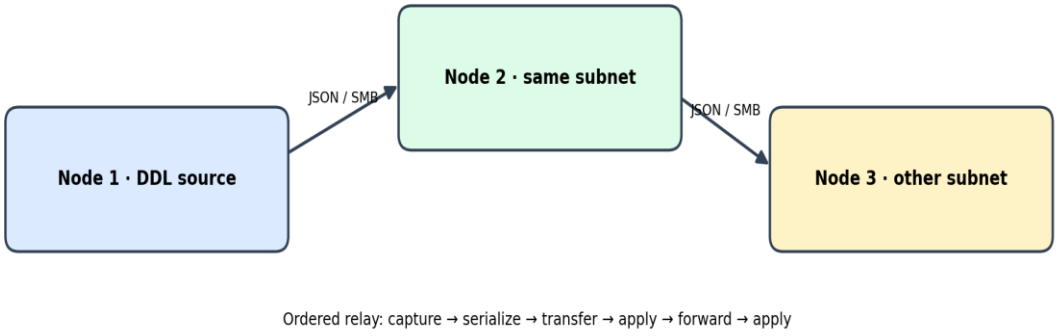


Figure 3*. Serial relay from Node 1 through Node 2 to Node 3 using JSON-over-SMB.*

The principal architectural difference is the Node 3 delivery path: direct from Node 1 in TCP/IP fan-out, but dependent on successful intermediate processing at Node 2 in the SMB relay. The additional serial stages are plausible contributors to the larger recorded relay delays, although their separate effects were not measured.

3.3 Native MySQL replication baseline

Native MySQL replication distributes database changes through MySQL's built-in logging and replica mechanisms [18,19]. It provides a practical baseline for comparing recorded synchronization delays but differs from application-level metadata propagation in its architecture, operational scope, and completion semantics. The precise replication mode, filtering configuration, and DDL completion criterion could not be established from the available experimental records. Consequently, native replication is treated as an operational performance comparator rather than a functionally equivalent experimental control.

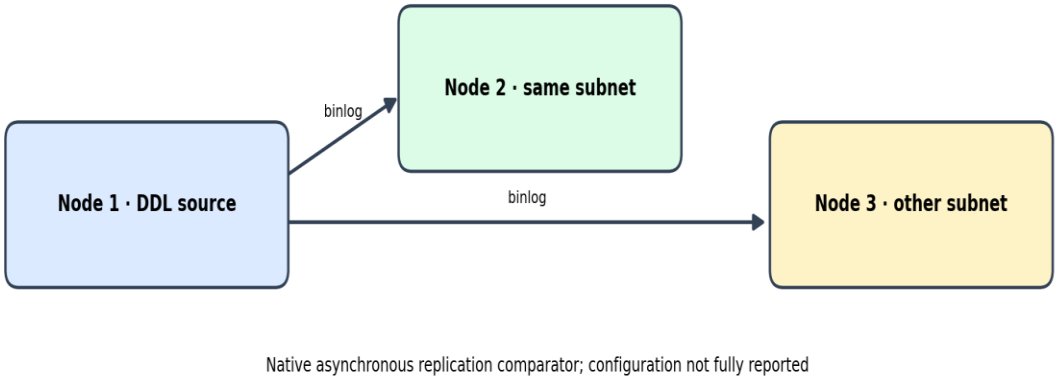


Figure 4. Native MySQL replication comparator; replication mode and completion criterion were not independently verified.

4 Measurement model and experimental methodology

For DDL event i and destination node j, let s_i be the source-detection timestamp, r_{ij} the destination-receipt timestamp, a_{ij} the destination-application-completion timestamp, and k_{ij} the acknowledgement-arrival timestamp. These identify distinct operational endpoints, rather than interchangeable measures of metadata consistency.

Event-delivery interval:

$$D_{ij}^{delivery} = r_{ij} - s_i \tag{5}$$

DDL application-completion interval:

$$D_{ij}^{application} = a_{ij} - s_i \tag{6}$$

Acknowledgement-arrival interval:

$$D_{ij}^{ack} = k_{ij} - s_i \tag{7}$$

For an event applied at all required destinations, the end-to-end application-completion interval is:

$$C_i^{application} = max_{j \in J_i}(a_{ij} - s_i) \tag{8}$$

J_i is the set of destinations required to apply event i. The maximum in Equation (8) represents the last required application completion, not a verified catalog-convergence time.

The 30 values recorded as delay_ms represent reported synchronization delays for the three complete implementations. Because the archived data do not consistently identify whether each recorded interval ends at receipt, DDL application, ACK arrival, or replica catch-up, these observations are not equated with any one of Equations (5)–(8).

A catalog-convergence endpoint would additionally require checking the expected destination schema states and the integrated global metadata version. Such an endpoint was not separately timed in the present experiment.

The source and destination MySQL node clocks were synchronized using the Windows Time Service to support cross-node timestamp recording for DDL transmission, reception, execution, and acknowledgement. Residual clock-offset measurements and uncertainty bounds were not separately retained; the reported cross-node delays should therefore be interpreted subject to possible timestamp uncertainty.

4.1 Experimental design and data provenance

The dataset contains five event-index observations for each of three implementations at each of two destinations (30 measurements). Node 2 is on the source subnet; Node 3 is on a different subnet. The retained records do not specify the original DDL statements, precise endpoint definitions, execution sequence, host settings, network RTTs, or independent trial repetitions in sufficient detail to verify workload matching or statistical pairing. The comparisons are therefore descriptive; no assumption of randomized trials or matched event pairs is made.

5 Recorded observations and descriptive statistics

Table 2. Recorded raw synchronization delay observations (ms); the 63 ms value is retained.

Implementation	Node 1 to Node 2 (same subnet)	Node 1 to Node 3 (different subnet)
Native MySQL replication	161, 149, 189, 167, 192	161, 149, 63, 167, 192
Direct TCP/IP fan-out	1113, 1015, 1338, 952, 1582	1346, 1178, 1519, 1130, 1742
Serial JSON-over-SMB	2569, 3584, 3794, 3464, 2794	4622, 4762, 5041, 4755, 4143

Table 3. Descriptive statistics; SD is the sample standard deviation and CV = 100 × SD/mean.

Implementation		Node	n	Mean	SD	Median	CV %	Range
Native MySQL replication		Node 2	5	171.6	18.5	167.0	10.8	149–192
Native MySQL replication		Node 3	5	146.4	49.2	161.0	33.6	63–192
Direct TCP/IP fan-out		Node 2	5	1200.0	259.0	1113.0	21.6	952–1582
Direct TCP/IP fan-out		Node 3	5	1383.0	252.4	1346.0	18.2	1130–1742
Serial JSON-over-SMB		Node 2	5	3241.0	530.2	3464.0	16.4	2569–3794
Serial JSON-over-SMB		Node 3	5	4664.6	329.1	4755.0	7.1	4143–5041

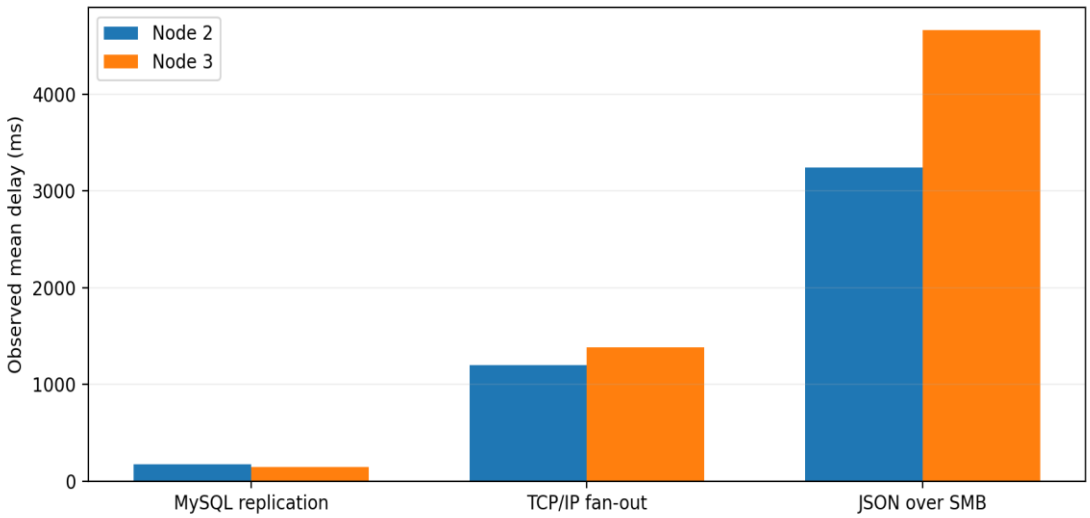


Figure 5. Observed mean delays for each complete implementation and destination (n = 5 per bar).

5.1 Architectural contrasts

At Node 2, the direct fan-out mean is 63.0% lower than the serial relay mean (1200.0 versus 3241.0 ms); at Node 3, the reduction is 70.4% (1383.0 versus 4664.6 ms). The serial/direct ratios are 2.70 and 3.37 respectively at node 2 and node 3. Relative to Node 2, the Node 3 mean changes by −14.7% for native replication, +15.3% for direct fan-out, and +43.9% for serial relay. The higher recorded latency at Node 3 is consistent with the additional forwarding and coordination stages of the serial relay architecture. However, the available measurements do not permit the individual contributions of subnet placement, SMB transfer, JSON serialization, or application-level processing to be isolated or quantified.

6 Event-level analysis and robustness

Table 4 reports every event index and its observed destination delay. Indices are not treated as paired experimental blocks until DDL identity and ordering are verified from logs.

Event index	MySQL N2	MySQL N3	TCP N2	TCP N3	SMB N2	SMB N3
1	161	161	1113	1346	2569	4622
2	149	149	1015	1178	3584	4762
3	189	63	1338	1519	3794	5041
4	167	167	952	1130	3464	4755
5	192	192	1582	1742	2794	4143

For direct fan-out, the observed latency ranges are 952–1582 ms at Node 2 and 1130–1742 ms at Node 3, below the corresponding serial relay ranges of 2569–3794 ms and 4143–5041 ms. These non-overlapping sample ranges describe the recorded runs, not guaranteed behavior for future workloads. Native replication at Node 3 has mean 146.4 ms, median 161 ms, and SD 49.2 ms. Its 63 ms observation is retained unchanged. Possible influences include DDL complexity, metadata locks, replication state, resource availability, network conditions, and residual clock-offset uncertainty. The event-specific records do not establish which factor caused this value.

The coefficient of variation (CV) complements mean latency by describing relative dispersion across the six implementation–destination combinations. Five observations per combination support descriptive comparisons but do not establish workload matching, trial independence, or a basis for causal protocol-specific claims.

Accordingly, the findings are reported as observed implementation-level differences, without inferential significance testing or confidence intervals.

7 Technical interpretation and correctness requirements

7.1 Latency decomposition and bottleneck hypotheses

Direct fan-out involves capture, queueing, serialization, transfer, destination-side execution, and ACK processing, with overall multi-destination completion governed by the last required successful destination. Serial SMB relay includes corresponding operations at two ordered hops and may add file handling, polling, and intermediate scheduling. Retry and recovery may add delay when invoked. The reported values are overall implementation-level observations; individual stage durations were not separately measured and cannot be estimated from the five values per implementation–destination combination. The decomposition therefore explains architectural latency sources rather than fitting a numerical stage-delay model.

7.2 Delivery, application and recovery semantics

The direct JSON-over-TCP/IP flow uses destination execution ACKs and retry following missing ACKs; the serial JSON-over-SMB flow incorporates failure detection, logging, retry, replay, dead-letter handling, and recovery. An ACK confirms reported successful processing when received, but a crash after DDL application and before durable completion recording or ACK delivery can leave its status ambiguous. Recovery must then distinguish a pending operation from one already applied to avoid unintended duplicate execution. Persistent event-state records, unique identifiers, controlled retries, and schema-state reconciliation are relevant safeguards. The latency-focused evaluation does not independently establish crash-safe exactly-once execution or verified global-catalog convergence.

8 Threats to validity and reproducibility

Internal validity is constrained by incomplete documentation of DDL workload matching, processing schedules, timing-endpoint equivalence, and residual inter-node clock offsets, despite use of the Windows Time Service. These factors limit causal attribution.

Construct validity depends on the meaning of the recorded delay_ms endpoint, which is not documented as equivalent across all three implementations. It should not be interpreted as a separate measurement of global metadata-catalog convergence.

External validity is bounded by three homogeneous MySQL nodes, two destination subnet placements, and five observations per implementation–destination combination.

Statistical generalization is limited by the small dataset and unavailable evidence of trial independence and workload matching. Native MySQL replication is an operational comparator, not a functionally identical control.

9 Future work and independent validation

Future work will focus on independent validation of the reliability and consistency properties specified in this study, rather than treating them as established by the present latency observations. A reproducibility package should document the DDL statements and initial schemas; MySQL and operating-system versions; hardware, network, and Windows Time Service settings; measured clock offsets and timing endpoints; TCP message framing and ACK/retry policies; SMB transfer and polling parameters; native replication configuration; source/destination event logs; and scripts reproducing the reported statistics and figures. Controlled experiments should use comparable DDL workloads, independently documented repetitions, and consistent stage-specific timestamps.

Fault-injection tests should examine node restarts before and after event receipt or DDL application, lost ACKs, duplicate deliveries, Node 2 relay interruption, network failures, and conflicting schema modifications. They should verify recovery, event ordering, duplicate prevention, destination schema equivalence, and global-catalog convergence. Additional work may evaluate unique event IDs, durable event-state records, ordered catalog versions, reconciliation after uncertain completion, and a defined global-catalog publication policy. Scalability tests should vary node count and event rate; a separate evaluation should assess the Calcite-based GUI and heterogeneous interoperability. These are future validation and extension commitments, not claims about completed experiments.

10 Conclusions

This study characterized DDL synchronization latency in a three-node homogeneous MySQL testbed using native replication, direct JSON-over-TCP/IP fan-out, and serial JSON-over-SMB relay. Native replication recorded the lowest mean delays at both destinations. Direct fan-out recorded means of 1,200.0 ms at Node 2 and 1,383.0 ms at Node 3, compared with 3,241.0 ms and 4,664.6 ms for serial relay—reductions of 63.0% and 70.4%, respectively. These differences characterize the complete implementations and do not isolate intrinsic transport-protocol effects.

The empirical contribution consists of 30 preserved observations, destination-level descriptive statistics, and event-index inspection. The architectural contribution distinguishes independent TCP/IP fan-out with ACK-based retry from ordered SMB relay with ACK chaining and failure/recovery handling. A formal global/local catalog-state specification complements the measured results, while the Apache Calcite-based GUI remains an architectural integration component.

The results provide a bounded empirical basis for selecting and refining DDL propagation architectures in homogeneous MySQL multidatabase environments. The distinction between observed latency, implemented reliability provisions, and formal catalog-consistency requirements is maintained throughout the study.

Declarations

Funding: No external funding was received for this research.

Conflicts of Interest: The author declares no conflicts of interest relating to this research.

Data Availability: All 30 recorded latency observations are presented in **Table 2**. The complete experimental logs and implementation configurations are not included in the manuscript.

Use of Artificial Intelligence Tools: ChatGPT was used to assist with language refinement, manuscript organization, and editorial presentation. The author retains responsibility for the accuracy of the experimental data, scientific interpretations, analyses, and conclusions.

References

[1] M. S. Tambe, “Bi-Directional Synchronization Engines for Heterogeneous Enterprise Content Platforms: A Conflict Resolution Framework Using Vector Clocks and Semantic Merge Policies,” *Computer Fraud and Security*, 2026.

[2] P. Mondal and E. Tilevich, “Cross-Platform Integration and Extension of Replicated Data Libraries in Distributed Systems,” *Journal of Parallel and Distributed Computing*, 2026. 10.1016/j.jpdc.2026.105263

[3] M. G. Ali, “Method and System Including Replication Agent Instances for Maintenance of Metadata in the Multidatabase System,” *Proceedings of IEEE ISNCC 2025*, 2025. DOI: 10.1109/ISNCC66965.2025.11250388

[4] M. Jasny, T. Ziegler, J. Nelson-Slivon, V. Leis and C. Binnig, "Synchronizing Disaggregated Data Structures with One-Sided RDMA: Pitfalls, Experiments and Design Guidelines," *ACM Transactions on Database Systems*, 2025. https://doi.org/10.1145/3716377

[5] P. Martins, A. S. da Silva, A. Afonso, J. Cavalcanti, and E. de Moura, "Supporting Schema References in Keyword Queries over Relational Databases," *IEEE Access*, 2023. DOI: 10.1109/ACCESS.2023.3308908

[6] C. J. F. Candel, D. S. Ruiz and, J. J. García-Molina, "Unified Metamodel for NoSQL and Relational Databases," *Information Systems*, 2022. 10.1016/j.is.2021.101898

[7] C. Giannoula, N. Vijaykumar, N. Papadopoulou, V. Karakostas, I. Fernandez and, J. Gómez-Luna, "SynCron: Efficient Synchronization Support for Near-Data-Processing Architectures," 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021. DOI: 10.1109/HPCA51647.2021.00031

[8] Federated Distributed Database Reference – SYNC SCHEMA Operations. https://docs.oracle.com/en/database/oracle/oracle-database/26/shard/federated-ddb-reference.html

[9] A. P. Sheth and J. A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys (CSUR)*, 1990. https://doi.org/10.1145/96602.96604

[10] M. T. Özsu and P. Valduriez, "Principles of Distributed Database Systems (latest editions)," *Springer Nature Link*, 2020. https://doi.org/10.1007/978-3-030-26253-2

[11] R. Elmasri and S. B. Navathe, "Fundamentals of Database Systems, fourth edition," *PEARSON*, https://www.uoitc.edu.iq/images/documents/informatics-institute/Competitive_exam/Database_Systems.pdf

[12] M. G. Ali and M. G. Murtuza, "Use of Component Integration Services in Multidatabase Systems: A Feasible solution for Integrating Academic Institutions or Commercial Industries," *Journal of Software Engineering and Applications*, 2023. doi: 10.4236/jsea.2023.1611029

[13] M. G. Ali, "Framework for Creating Global Schema Using Global Views from Distributed Heterogeneous Relational Databases in Multidatabase System," *Global Journal of Computer Science and Technology*, 2010. https://computerresearch.org/index.php/computer/article/view/918

[14] M. G. Ali, "A Framework for Creating Global Schema Using Proxy Tables from Distributed Heterogeneous Relational Databases in Multidatabase System," *International Journal of Computer and Electrical Engineering*, 2010. http://www.ijcee.org/show-34-608-1.html

[15] M. G. Ali, "Object-Oriented Approach for Integration of Heterogeneous Databases in a Multidatabase System and Local Schemas Modifications Propagation," *(IJCSIS) International Journal of Computer Science and Information Security*, 2009. https://doi.org/10.48550/arXiv.0912.0603

[16] M. G. Ali. "A Multidatabase System as 4-Tiered Client-Server Distributed Heterogeneous Database System," *(IJCSIS) International Journal of Computer Science and Information Security*, 2009. https://doi.org/10.48550/arXiv.0912.0579

[17] M. G. Ali, "Local Schemas Modifications Propagation and Maintenance of Metadata in Multidatabase System," *International Journal of Computer Applications*, 2010. 10.5120/1112-1456

[18] MySQL 8.4 Reference Manual, Chapter 19: Replication. https://dev.mysql.com/doc/refman/8.4/en/replication.html

[19] Evaluation of Database-Level Replication and Binary Logging Options. https://dev.mysql.com/doc/refman/8.4/en/replication-rules-db-options.html